GIER SYSTEM LIBRARY

Order No.: 494

Class: 0.2.2

Type: Book

Author: T. Asmussen, J. Jensen,
S. Lauesen, P. Lindgreen,
P. Mondrup, P. Naur, and
J. Zachariassen

Ed.: December 1967 (E)

THE COMPLETE ANNOTATED PROGRAMS OF GIER ALGOL 4

Volume I

Tape identification	Contents	Page identification	Number of pages
T1,L1	Block head	T1, L1 Gier Algol 4	1
	- -	check load parameters	1
	General pass adm.	GP, page 3-13	11
	Running system	RS, - 1-18	18
T2	Tape test	T2 Gier Algol 4	1
	Pass 1	pass 1, page 1-35 11, 14, 22 (intended for comments) missing	32
	Pass 2	pass 2, page 1-6	6
	Pass 3, phase 3.1	pass 3 (phase 1) (std proc) page 1-3	3
	Sum check	tape number := 3	1
T3	Tape test	T3 Gier Algol 4	1
	Pass 3, phase 3.2	Pass 3, page 1-22	22
	Pass 4	pass 4, page 1-12	12
	Pass 5, phase 5.1	pass 5, page 1-11	11
	Pass 5, phase 5.2	pass 5 std. proc descr., page 1-3	3
	Sum check	tape number := 4	1

REGNECENTRALEN

```
;slip<

[31.8.67                                                    T1, L1 Gier Algol 4
                                                            1]

b c82, e69 ; Outermost block


[Definition of loading parameters
          Possible values    , meaning]

e14=90    ; Depending on available drum, first track for reading by slip.
e4=15     ;     10 < e4  < 17  , first core used by translator.
e20=1022  ; 1005e4 ≤ e20 ≤ 1022, last   -    -   -      -
e38=15    ;     10 ≤ e38 ≤ 200 , first core used by translated program.
e37=1022  ;    800 ≤ e37 ≤ 1022, last   -    -   -      -         -
          ;                     Note below: e37 = 1e37.
e27=0     ;   0, arrays in core, e18 and e40 must also be 0.
          ;   1,   -    - buffer.
e44=0     ;   0, translator medium is drum.
          ;   1,     -        -    - a buffer medium.
e18=0     ;   0, backing store is drum.
          ; 400,    -      -   -  disc with block length 400.
          ; 640,    -      -   -   -    -    -    -   640.
e40=0     ; < 0, no tape units , SAMBA record   handling procedures are included.
          ; = 0, -   -     -   ,   -     -          -        -       - not  -   .
          ; > 0, e40 -     -   ,   -     - and tape -        -       -      -   .
e69=4     ; 4 ≤ e69 ≤ 40 , SAMBA record segment length.

i redefine wanted loading parameters
s

[here follows stop code and clear code]


<e40+1,<-e40+1
×
e27=1
>
```

```
[check load parameters]
<e4-17,e4=0>
<-e4+10,e4=0>
<e20-1022,e4=0>
<-e20+1005e4,e4=0>
<e38-200,e4=0>
<-e38+10,e4=0>
<e37-1022,e4=0>
<-e37+800,e4=0>
<e27-1,e4=0>
<-e27,e4=0>
<e44-1,e4=0>
<-e44,e4=0>
<e40-15,e4=0>
<-e69+4,e4=0>
<e69-40,e4=0>
<-e27+1,<-e18,e4=0>
<-e27+1,<e18,e4=0>
e3=1
<e18+1,<-e18+1,e3=0>
<e18-399,<-e18+401,e3=0>
<e18-639,<-e18+641,e3=0>
<e3,e4=0>
<-e4+1
i wrong redefinition, restart the slip reading
e
s

>


[define some general names]
e13=21e4      ; start translator buffer areas
e28=163e13    ; start adress of GP fixed part
e3=40e28      ; byte input-output code
e16=40e3      ; normal first word of a pass
e37=1e37      ;
e41=1         ; GP seg size
<e44          ; if translator medium is buffer medium then
e41=40        ;   GP seg size := 40;
>             ;
e26=41        ; trackplace length for RS

[version and tape number definitions]
d e48=0       ; translator is in := false
d e51=1       ; version number
d e50=e51     ; max version number := version number
              ; set version number for :
d e61=e51     ; T1,L1
d e62=2       ; T2
d e63=3       ; T3
d e64=4       ; T4
d e65=5       ; T5
d e66=6       ; T6,L2
d e67=7       ; T7,L3
d e68=8
```

```
b k=e14, i=e28, a25,b35,c7 ; GP drumblock. GP fixed part, track e14.

e28: qq [loaded again at end GP] ; start compiler: goto init GP;
[164e13]                         ;
     qq [loaded again at end GP] ;
a16:
a:   1₁₀-3                       ; constants:
[1a] 1                           ;
[2a] 10                          ;
[3a] 960                         ;

b1:
e6:  bt474[print count]t-55; test print: print count:= print count + 1;
     hs  a4                  ;   if print count > 10 then newline;
e7:  gr  e13   , gm  41e13   ; print: save R:= R; save M:= M;
     ck  0     , tl  -69     ;   RM:= part 1(R);
e8:  mln a     , pt  b2      ; print 1: RM:= RM × ₁₀-3; zero:= 0;
a1:  ck  -10   , ga  b3      ; next: char:= Raddr:= part 4(R);
     bs (b4)           LZ    ;   if digits = 3 ∨ R ≠ 0 then zero:= 16;
     pt  b2    t   16       ;
b2:  pa  b3    t [zero] LZ   ;   if R = 0 then char:= zero;
b3:  sy [char] , ncn sa10    ;   outchar(char); if -, call from byte out then
a2:  pa  b1    t   474       ;     print count:= 10;
b4:  ncn 0[digits]t-256      ;   digits:= (digits + 1) mod 4;
     mln 2a    , hv  a1      ;   if digits ≠ 0 then begin R:= RM × 10 goto next end;
     arn e13   , pm  41e13   ;   R:= save R; M:= save M; marks:= marks A;
a3:                          ; space exit:
b5:  sy  59 [point], pa b5   ;   outchar (point); point:= 0;
     sy  0     , hr  s1      ;   outchar(0); return;
a4:                          ; new line: outchar(64) print count:= 0
e9:  sy  64    , pa  b1      ;   out char (passno); passno:= 0;
b6:  sy 1[passno], pa b6     ;   goto space exit;
e29h:hv  a3    , it  a17     ; next segm: GPseg:= next seg 1; goto set GP pl;
e5:  pa  b11   , it  123e13  ; mess: GPseg:= mess1; set GP pl: GPpl:= if outaddr >
     bs (b15)  , it  83e13   ;     outbuf 2 .. (if backw pass then 1 else 0) then
b10:pa124e13[GPpl]DXt124e13; ;     outbuf 1 else outbuf 2; swop;
e42: hs  b11                 ; next GP seg: get GP seg; goto GP pl;
e30:                         ;
b11: arn a19[GPseg] Dt e41   ; get GP seg: GP seg:= GP seg + GP seg size;
< e41-1                      ;   if GP in buf then
     ck  10    , ar  11e4    ;
     ar (b10)  D             ;     buf to core (40) words from: (GP base + GP seg)
e43: qq 0[segno] , vk (b9)   ;       to: (GP pl);
     il  0     , hv (s-1)    ;
×                            ;   else
     hs  a5                  ;     begin
     lk (b10)  , vk (b9)     ;     track (GP base + GP seg); lk (GP pl);
e43: qq 0[segno] , hv(s-1)   ;     end;
a5:  ck  10    , ar  11e4    ;   wait drum; go indirect (s-1);
>                            ;
e11: gm  41e13 X             ; track: save M:= M;
     dln 3a    , ck  -10     ;
     ga  b8    , cl  -10     ;   group:= 960 + R : 960;
     ga  b9    , pm  41e13   ;   the track:= R mod 960; M:= save M;
b8:  vk [group]t   960       ;   select group;
e17:
b9:  vk[the track], hr s1    ;   select the track; return;
e3: d e22=1, e24=0, e10=i    ;   Define GP loading parameters
```

```
b c7                                ; mess 1, track 1e14

c1:  hrn r1     X                   ; message: restore s from original call; M:= 0;
     arn s1     , it 41e13          ; R:= paramword in s1;
     bs (b17)   , it  -41           ;   text:= GPpl:= if inaddr ≥ inbuf 2 then inbuf 1
     it (rb21)  , pa  b10           ;     else inbuf 2;
     gt  rb21   , cl  -2            ;   rel:= part 2(R);
     tk  38     , ck  -18           ;   by no:= bits(36, 37, R); M pos 2:= bits(38, 39, R);
     ac  r, ud13e4[byaddr]          ;   do(select by no);
     sy  58     , arn e4            ;   write LC;  R:= M;  M:= linecount;
     hs  e9     X                   ;   new line;
     nc  0      , gs  rb23          ;   if Raddr ≠ 0 then return addr:= s;
     ca  3.2    , hv  rc2           ;   if Raddr ≠ 3.2 then
     arn 16e4   , ab  rb22          ;     begin  error occurred:= true;
     ga  16e4   , sy 29[RED];       ;       write char (RED)  end;
c2:  sy  35[l]  , sy  57[i]         ;   writetext (≪line≫);
     sy  37[n]  , sy  53[e]         ;

c3h: qq  rc5    , hsn e8            ;   print 1 (CR count)EP seg := 0;
                                    ;     comment to get this GPseg again;
c4h: pa  b11    , hs  e30           ; next text segm: get GP seg;
c5:                                 ;
b21: pmn42e13[text]Xt [rel];        ; next text word:
     pt  rb21   t   -40             ;   text:= text + rel; rel:= -40
     is (b10)   , it  s39           ;   if text > GPpl + 39 then goto next text segm;
     bs (rb21)  , hh  rc4           ;
                                    ;
c6h: cl  34     , ck  -4            ;   write text (word[text]);
     ga  rb22   , ca  15            ;   if more words then
     hv (rb21)  Dt   41             ;     begin text:= text + 41; goto next text word end

b22: sy1.3[char], cln -6            ;
     [see 3b20]                     ;
     nc  160    , hh  rc6           ;
     pa  b11    t   a19             ;   GPseg:=  end comp;
     arn 8e4    , hs  e11           ;   from drum (input track, GP pl);
     lk (b10)   , sy  62            ;   writechar (BLACK);
b23: ps e42-2   , ud  16e4          ;   s:= return addr;  select(normal;
     [return addr]                  ;
[-2] grn rc1-1 VX  1    M           ;   clear this track for marks
     hr  s1                         ;   return; comment if stop then to
     hv  r-2                        ;     next GP seg which will be end comp;

d e31=k, e32=j                      ; Define text loading parameters
d e10=120e10, e22=3e22              ; Reserve room for mess 2 and 3, tracks 2 and 3 e14.
d e24=e24+e41+e41+e41               ;

e                                   ; end mess 1
```

```
b k=e31, i=0                    ; load common texts.
t=e32                           ;
e33: tprogram too big;          ;
e34: tstack;                    ;
e45: tpass sum;                 ;
e46: tpass medium;              ;
e32=i;
e    ;
```

```
b c11                                  ; Block for next seg 1, track 4e14.
i=e10, a17=e24                         ; Define GP seg for next seg 1.
e23=k                                  ; Define abs track for segment table
c1:  it (e43)  t   1                   ; next segment: seg no:= seg no + 1;
     arn rc3   X          IRC ;   R:= M; M:= set RC (seg table[seg no]);
     hh  rc10             NZ  ;   if R ≠ 0 then goto segment driver only;
     hh  rc9              LRB ;   if LRB then goto segment only;
c2:  arn rc4   t   1      IRC ; prepare new pass:
     gr  e13   t   1      MRC ;   The code from 1c4 to c10 is moved to inbuf1 ff,
     hv  rc2              NB  ;   this place is cleared for marks and
     hv (rc11) DVt c5         ;   the moved code is entered.
[segment table]               ;
c3: e21=j, a8=c3-e28           ; Define relative addresses for segment words.
    ; Format
qq  ; GP, not segment word
qq  ; 2   pass 1          Each segment of the compiler is described by one word
qq  ; 1   pass 2          which is loaded by the segment itself or, for the
qq  ; 1   pass 3.1        std pr segments,  by merger.
qqf,; 3   std.pr.ident.    Three main formats are used:
qq  ; 2   pass 3.2
qq  ; 1   pass 4          1. Words which describe a new pass, i.e. byte input-
qq  ; 1   pass 5.1           output is adjusted:
qq  ; 2   pass 5.2           qq first core.9 + words.19 + pass no.24 + pass bits.29 +
qqf,; 2   std.pr.descr.                                        entry in core.39;
qq  ; 1   pass 6             pass bits: change direction: 1.29 else 0.29
qq  ; 1   pass 9                        backward pass   : 1.28 else 0.28
qq  ; 1   pass 7
qq  ; 1   pass 8.1        2. Words which describe next part of current pass:
qqf,; 2   std.pr.code        qq first core.9 + words.19 + 1.20 + entry in core.39 f;
qq  ; 2   pass 8.2
qq  ; 2   pass 8.3 (RS)   3. Words which describe a std.proc segm:
    ;                        qq words.19 f,;

c4=i-1, c5=1e13-e30            ; terminate byte output
     pi (16e4) , hsn e3        ;   in:= modebits; byte out (0);
     grn a9    , hsn e3        ;   prevent testoutput; byte out(0); byteout(0);
c6:  pp  rc7   , hsn e3        ;   change return from full track to c7;
     gp  b16   , hh  rc6       ;   while track not full do byte out(0);

c7:  hh  rc9           NPA ; if pass inf wanted then
     gm  rc7   , hs  e9    ;   begin newline; print 1(used tracks);
     pmn 1e4   , hs  e8    ;   if inf 1 ≠ 0 then print (inf 1);
     arn 2e4   , pm  3e4   ;   if inf 2 ≠ 0 then print (inf 2)
e8:  nc  0     , hs  e7    ;   end;
     hvn rc8   X     NZ    ; Segment only: set return to get GP seg;
c9h: pm  rc7   , ps  e42   ;   goto next GP seg;

c10h:hvf e42   , ga  b10   ; segment driver only: GP p1:= Raddr;
[-2] grn rc3   Vt  1   M   ;   clear this place for marks;
c11: hr  e30               ;   comment also used from c2 ff;
[1]  hv  r-2               ;   return to (get GP seg);
e
e10=40e10, e22=1e22            ; Set GP load parameters
e24=e24+e41                    ;
e10: [check load addr]         ;
```

```
b c11                          ; Common block for next seg 2, track 5e14
                               ;   only one of the following versions is loaded:

< -e44+1                       ; Drum version:

c1:  gmn e13   XV      IZC     ; Get segment: e13:= R:= M; goto start;
c2:  arn 12e4  , hv  rc7       ; Get word: R:= track rel pos 9 + track no pos 39;
                               ;   goto move word;
     gt  r     , pp [words]    ; Start: sumok; transport ok:= t;

c3h: ga  rb25  , arn 12e4      ; next: R:= track rel pos 9 + track no pos 39;
     bs  p-512 t   -473        ;   if 39 < p ∧ trackrel = 0 then
     ca  0     , hv  rc6       ;     goto whole tracks;
     hs  rc7                   ;   move word;
     pp  p-1   , it   1        ;   p:= p - 1; first core:= first core - 39;
c4:  qq (rb25) t   40          ; after whole tracks: first core:= first core + 40;
     ncn p     , hh  rc3       ;   if p ≠ 0 then goto next;
     pa  rb26  , pmn 8e4       ;   new:= true;
     hs  rc9   X               ;   fill buf (input track);
c5:                            ;
b25: ar e16-1[firstcore]t-1    ; sumit:
     ar  2     D       LA      ;   R:= sum of read words including marks;
     ar  1     D       LB      ;   M:= e13;
     pm  e13   X               ;   sumok:= R = 0;
     ca (rb25) X               ;   return;
     hr  s1            IZB     ;
     hv  rc5                   ;

c6:  hs  rc10                  ; whole tracks:
     lk (rb25)                 ;   from drum (first core, trackno);
     pp  p-40  , arn 1a16      ;   p:= p - 40; trackno:= trackno + 1;
     ac  12e4  , hv  rc4       ;   goto after whole tracks;
c7:                            ;
b26: bs [new]  , hh  rc8       ; move word:
     hs  rc9                   ;   if new then fill buf(trackno);
c8h: arn 12e4  , ga  rb27      ;   rel:= trackrel;
     ar  1     D               ;   track rel:= track rel + 1;
     ca  40    , ar  rc11      ;   if trackrel = 40 then
     ga  rb26  , it (rb28)     ;     begin track rel:= 0; trackno:= trackno + 1 end;
b27: pm [rel]  X       IRC     ;   new := track rel = 0;
     gr (rb25)         MRC     ;   store[first core] marks := R:= store[rel+trackbuf];
     gm  12e4  , hr  s1        ;   return;

c9:  hs  rc10                  ; fill buf:
     pa  rb28  , it  41e13     ;   track buf:=
     bs (b17)  , it  1e13      ;     if inbuf 2 < inaddress then inbuf 1 else inbuf 2;
b28: lk[trackbuf] t 42e13      ;   from drum (track buf, bits (10, 39, R));
     vk (e17)  , hrn s1        ;   wait drum; R:= 0; return;

c10: tk  10    , ck  30        ; subroutine for track select;
     hv  e11                   ;

c11: qq  -40   t   1.29        ; constant used in 2c8;
```

```
×                                   ; Buffer version:
                                    ;
c1:  gmn e13   XV      IZC          ; Get segment: e13:= R:= M; M:= 0; goto start;
c2:  arn 1a16  , hv  rc3            ; Getword: R:= 1 goto
     gm  rb26  , ga  rb25           ; start: get word exit:= false; first core:= bits(0, 9, R);
     ck  10    , tl  -30            ;   R:= bits(10, 19, R); sum ok:= transport ok:= t;
c3:  sr  12e4  , gr  41e13          ; get R words: words:= R;
     qqn                NT          ;   if rest in buf < R then R:= rest in buf;
     sc  41e13 , ar  12e4           ;   words:= words - R;
     hh  rc7            LZ          ;   if R = 0 then goto get next block;
     tk  20    , ar  rb33           ;   il (R) words from: (topbuf - restinbuf) to:
     ar (rb25) D                    ;     (firstcore);
     sr  12e4  , il  0              ;
b26: arn 1a16  , hv  rc6            ;   if get word exit then goto exit 1;
     ck  20    , tk  30             ;   first core:= first core + R;
     ac  rb25  , ck  10             ;   rest in buf:= rest in buf - R;
c4h: sc  12e4  , arn 41e13          ; test more words: R:= words;
     hv  rc3            NZ          ;   if R ≠ 0 then goto get R words;
c5:
b25: ar e16-1[firstcore]t-1         ; sumit:
     ar  2     D        LA          ;   R:= sum of read words including marks;
     ar  1     D        LB          ;   M:= e13;
     pm  e13   X                    ;   sum ok:= R = 0;
     ca (rb25) VX                   ;   return;
c6:  sc  12e4  , arn e16-1          ; exit 1: rest in buf:= rest in buf - 1;
     hr  s1             IZB         ;   R:= store[first core]; return;
c7h: hv  rc5   , pa  rb29           ; get next block: err ct:= 0;
     arn rb30  , ac  rb31           ;   curr block:= curr block + block in cr;
c8:                                 ; repeat block:
b27: arn rb31  , il [unit]          ;   il block (curr block, unit);
b28: arn rb34  , il [check]         ;   il status (check word, check);
     il  0     , pm  rb32           ;   us (1) word from: (addr of curr block) to:
c9:  ar  rc10  D                    ;     (addr of curr block in buffer);
     us  0     , arn 12e4           ;   if status < 0 then
b29: bt [errct]Vt-200  LT           ;     begin err ct:= err ct + 1; if err ct < 3 then
     gm  12e4  , hh  rc4            ;       goto repeat block;
     hr  s1             IZC         ;     sumok:= transport ok := f return
     hv  rc8                        ;     end;
     qq  [fill]                     ;   rest in buf:= block length; goto test more words;
b30: qq [block incr]                ; These parameters are set by
b31: qq [current block]             ;   init buf compiler
b32: qq [block length]              ;
b33: qq [top buf]                   ;
b34: qq 12e4.9+1.19+b31.39-e28.39   ; word for transport of status word and
                                    ;   curr block, top buf is added by init buf comp
c10=b31-12e4                        ; address diff. for transport word;

>                                   ; End buffer version
e                                   ;
e10=40e10, e22=1e22                 ; Set GP load parameters
e24=e24 + e41                       ;
e10: [ check load address]          ;
```

```
b b3, c5                    ; Block for next seg 3, track 6e14

c:   cln -20   , ga  rb1    ; Unpack rest of segment word from M:
     gt  rb2   , tl -5      ;   pass bits:= bits(20,29,M);
     ps  e13-1 , pm  rc5    ;   exit:= bits(30,39,M); Raddr:= bits(20,24,M);
     cl  -10          NZA   ;   if -, sum ok then mess(if -, transport ok
b1:  pi[passbits]Vt508 LZB  ;     then ‹<pass medium› else ‹<pass sum›, 2, 0);
     gm  e13   , hv  e5     ;   comment mess is called in a funny way because
     hh  rc4          LZA   ;   it overwrites this track; pass bits 089 to in;
d c6=e20+i-c-40             ;   if LZA then goto possible pause;
     ca r-c6-i+9,hhn e29    ;   if this place is at e20-40 then
                            ; skip pass 9: goto next segm;
     ga  b6    , ud  14e4    ;   pass no:= Raddr; select(typewriter);
                            ; prepare new pass:
     pm  8e4   , arn 9e4    ;   M:= input track; R:= output track;
     hv  rc1   X      LRA   ;   if -, change direction ∧ discmode then
     hv  rc1          NQA   ;     begin R:= last bit (R - first track);
     sr  5e4   , mb  1a     ;     M:= R + first track; R:= first track - R - 1
     gr  8e4   , ar  5e4    ;     end
     pm  5e4   X            ;   else if change direction then swap;
     sr  8e4   , sr  1a     ;   input track:= M; output track:= R;
c1:  gm  8e4   , gr  9e4    ;
     qq (b11)  t  e41  LRB  ; Get byte input output segm:
     qq  r2    , it  e3     ;   if backward pass then GP seg:= GP seg + GP segsz;
     pa  b10   , hs  e30    ;   GPpl:= byte out; get GP seg;
     pa  2e4   , pa  3e4    ;   inf 1:= inf 2:= 0;
c2:  arn 8e4   , hs  e11    ;   track(input track);
     pa  b5    t   59       ;   point:= 59; print count:= 10;
     lk  1e13  , ud  a2     ;   from drum(the track, base inbuf 1)
     arn 8e4   , ar  1a     ;   if backward pass ∧ discmode then
     grn e4    V      NRB   ;     output track:= input track + 1
     gr  9e4          LQA   ;   else if -,backward pass then CRcount:= 0;
     grn a12          LQA   ;   if discmode then no release used tracks;
     grn 1e4          LQA   ;   if discmode then used tracks := 0;
     arn a9    , gt  e5     ;   set test value to find GP pl;
c3:  arn(b18)  t   1        ;   while next byte = 0 do;
     hs  a14          LA    ;   byte addr:= byte addr - 1
     qq (b18)  V   -1  NZ   ;
c4h: hv  rc3   , ud  14e4   ; possible pause: select (typewriter);
     lyn       D      LKA   ;   if LKA then lyn;
     pi (16e4) , ud  16e4   ;   in:= mode bits;  select(normal);

[-2] grn rc-1  Vt  1   M    ; finis: Clear this place for marks;
b2:  hv  0     t  [exit]    ;   goto exit;
     hv  r-2                ;

c5:  qqn e46   t e45+1.26   ; packed text params for use if -, sumok;
e                           ; end next seg 3

e10=40e10, e22=1e22         ; Set GP load parameters
e24=e24+e41                 ;
e10: [check load addr]      ;
```

```
b k=e22+e14, i=e3, a1, b1  ; Forward pass track, track 7e14, core e3
a18=e24 [GP seg forward]   ;   comment must follow next seg 3;

e3:  hv 1e3[i] t   2   NKB ; byte out: comment 1e3 corresponds to i = 0;
a9:  a10=-a9[used in a2-1]       ;
     hs e6 t 123e13 [see endpass] ;   if LKB then test print;
     hv (e3)   t   2       ;   i:= i + 1; goto pack [i];
e12:                       ;
b15: gr 82e13[out addr] t 1; pack[1]: out addr:= out addr + 1;
     hr  s1                ;   buf[out addr]:= R; return;
     ck  -10   , gt (b15)  ; pack[2]: part 2 buf [out addr]:= Raddr;
     hr  s1                ;   return;
     ck  20    , ac (b15)  ; pack[3]: R:= R shift 20;
     hr  s1                ;   buf[out addr]:= buf[out addr] + R; return;
     ck  10    , ac (b15)  ; pack[4]: R:= R shift 10;
     pa  e3    t   a9      ;   buf[out addr]:= buf[out addr] + R; i:= 0;
     hr  s1            NA  ;   if -, mark A then return;
     arn 10e4  , ac  1e4   ; track out: used tracks:= used tracks + increm;
     arn 4e4   , sr  1e4   ;   if used tracks > available tracks then
     hs  e5            LT  ;     mess (<<program too big>, 2, 0);
     hv  r1    , qqn e33   ;
     hs  a1                ;   next track (output track);
     qq  9e4   , arn(b15)  ;   R:= buf[out addr];
     is (b15)  , sk  s-40  ;   to drum (the track, out addr - 40);
     qq (b15)  t -82   LB  ;   if mark B then out addr:= out addr - 82;
b16: hv  e12 [see end pass];   goto pack[1];
     [a12 is cleared by end pass when discmode]
a12: srn 10e4  , ac  1e4   ; track in: if -, discmode then
     hs  a1                ;     used tracks:= used tracks - increm;
     qq  8e4   , arn(b1)   ;   next track(input track); R:= buf[in addr];
     is (b1)   , lk  s-40  ;   from drum (the track, in addr - 40);
     qq (b1)   t -82   LB  ;   if mark B then in addr:= inaddr - 82
e2:b1:a14:
b17: arn 81e13 [inaddr] t 1; unpack next: in addr:= inaddr + 1;
[1e2]ga  e13-4 V       NA  ;   R:= buf[in addr];
     [NA is removed by init comp for use in type in in pass 1]
     hv  a12               ;   if mark A ∧ -, pass 1 then goto track in;
     tk  10    , ga  e13-3 ;   for j:= 1 step 1 until 4 do
     tk  10    , ga  e13-2 ;     byte buf[j]:= part (j) of: (R);
     tk  10    , ga  e13-1 ;
b18:                       ;
e1:  arne13-1 [byteaddr]t-4;   byte addr:= byte addr - 4;
     hr  s1                ;   R:= byte buf [byte addr]; return;
a13:                       ;
a1:  arn 10e4  , ac (s1)   ; next track: comment addr of track no in s1;
                           ;   track no:= track no + increm
e15: [pass 1 starts here and replaces the next 4 words by
     arn(s1)   y hv  e11   ;   R:= track no; goto track
     ...                   ;   ...                    ]
     arn 6e4   , sr (s1)   ;   if track no > last track then
     arn (s1)  V       NT  ;     track no:= track no - available tracks;
     srn 4e4   , hh  a1    ;   R:= track no; goto track
     hv  e11               ;
     qq [fill]             ;
d e24=e24+e41, e10=40e10   ; Set GP load parameters
d e22=1e22                 ;
e                          ;
```

```
b k=e22+e14, i=e3, a1      ; Backward pass track, track 8e14, core e3;

e3:  hh  e3[i] t   2   NKB ; byte out: comment e3 corresponds to i = 0;
                           ;    if LKB then test print;
     hs e6 t 122e13 [see endpass];   i:=  i + 1; goto pack[i]
     hv  a1    , ck  10    ; pack[1]: R:= R shift 10;
e12:                       ; store first: out addr:= out addr - 1;
b15: gr123e13[outaddr]t-1  ;   buf[out addr]:= R; return;
     hr  s1    , ck  20    ; pack[2]: R:= R shift 20; buf[out addr]:=
     ac (b15)  , hr  s1    ;      buf [out addr] + R; return;
     ck  -10               ; pack[3]: part 2 buf [out addr]:= Raddr;
     gt (b15)  , hr  s1    ;   return;
     ac (b15)              ; pack[4]: buf [out addr]:= buf [out addr] + R;
     pa  e3    V   e3      ;  i:= 0;
a1:  hh (e3)   t   2       ; comment a1 used after test print;
     hr  s1           NA   ;   if -, mark A then return;
e25: arn 10e4  , ac  1e4   ; track out: used tracks:= used tracks + increm;
     arn 4e4   , sr  1e4   ;   if used tracks > available tracks then
     hs  e5          LT    ;     mess (<<program too big>>, 2, 0);
     hv  r1    , qqn e33   ;
     hs  a13               ;   next track (output track);
     qq  9e4   , arn(b15)  ;   R:= buf[out addr];
     is (b15)  , sk  s1    ;   to drum (the track, out addr + 1);
     qq (b15)  t   82  LB  ;   if mark B then out addr:= out addr + 82;
b16: hv  e12 [see end pass];   goto store first;
     [a12 is cleared by end pass when discmode];
e35:
a12: srn 10e4  , ac  1e4   ; track in: if -, discmode then
     hs  a13               ;     used tracks:= used tracks - increm;
     qq  8e4   , arn(b17)  ;   next track (input track); R:= buf[in addr];
     is (b17)  , lk  s1    ;   from drum (the track, in addr + 1);
     qq (b17)  t   82  LB  ;   if mark B then in addr:= inaddr + 82;
e2:
b17: arn 42e13 [inaddr] t-1; unpack word: in addr:= in addr - 1;
     ga  e13-1 V       NA  ;   R:= buf[in addr];
     hv  a12               ;   if mark A then goto track in;
     tk  10    , ga  e13-2 ;   for j:= 4 step -1 until 1 do
     tk  10    , ga  e13-3 ;     byte buf[j]:= part (j) of: (R);
     tk  10    , ga  e13-4 ;
e1:  arne13-1[byteaddr] t-4;   byte addr:= byte addr - 4;
     hr  s1                ;   R:= byte buf [byte addr]; return;

a13: srn 10e4  , ac (s1)   ; next track: comment addr of track no in s1;
     arn(s1)   , sr  5e4   ;   trackno:= trackno - increm;
     arn(s1)   V       NT  ;   if trackno < first track then
     arn 4e4   , hh  a13   ;     trackno:= trackno + available tracks;
     hv  e11               ;   R:= track no; goto track;
     qq [fill]             ;
e16:                       ; first instruction of a pass is read in here;

d e24=e24+e41, e10=40e10   ; Set GP load parameters
d e22=1e22                 ;
e                          ;
```

```
i=e10                                ;
b c7                                 ; Block for init comp, track 9e14;
a19=e24                              ; Define GP seg for init comp;

     bs (e43)  , hv  rc6   ; End comp or init comp drum version:
     arn 11e4  , ar  rc5   ;   if segno > 0 then goto end comp;
                           ;   trackno:= base GP + tracks in GP; track rel:= 0;
a21: gr  12e4  , pm  rc4   ; Init comp buffer or paper tape vesion:
c1h: pp  86    , pp  p-1   ;   for i:= 39 step -1 until 0 do .....
     can p-4   , it  -43   ;     inbuf 2[i] mark0:= outbuf 1[i]mark0:= 0;
     grn123e13 t   -1  M   ;   for i:= 4 step -1 until 0 do
     bs  p-13  , hh  rc1   ;     byte buf[i] mark0:= 0;  ..
     acn 13e4  t   -1  M   ;   for i:= 12 step -1 until 0 do
     bs  p-4   , hh  rc1   ;     marks of e4 params:= 0; ..
     gm  17e4  t   -1  MA  ;   for i:= 3 step -1 until 0 do
     bs  p-2   , hh  rc1   ;     e4 param[i+13]:= instr(qq 1023, vy);..
     grn 205e13t   -82 MA  ;   buf limit 3 mark A := buf limit 1 mark A:= 0;
     grn 164e13t   -82 MC  ;   buf limit 2 mark C:= base in mark C:= 0;
     bs  p-1   , hh  rc1   ;

     pa  b11   t   a18     ;   GP seg:= forward pass; GP pl:= byteout;
     qq  r2    , it  e3    ;   get GP seg;
     pa  b10   , hs  e30   ;
[2]  arn 8e4   , ga  15e4  ;   save return point addr in 15e4
     tk  14    , gr  2e4   ;   Save return point track in pos 25 in 2e4;
     arn 6e4   , gr  -2    ;   Save search and start medium track in -2;
                           ;   ...  ....  .....  ..
c2h: arn 9e4   , pp  p-1   ;   for i:= 1 step -1 until -2 do
     ck  -10   , gt  p16e4 ;     by select[i]:= part [i, units];
     bs  p3    , hh  rc2   ;
     grn e4    , arn 5e4   ;   CRcount:= 0;
     gr  9e4   , ar  4e4   ;   output track := first track;
     sr  1a16  , gr  6e4   ;   stringtrack:= inputtrack := lasttrack:=
     gr  8e4   , hs  e11   ;     first track + available tracks - 1;
     sk  42e13 , pi (10e4) ;   to drum(string track, inbuf 2);
     gi  16e4  , arn 1a16  ;   modebits:= part 1(param 10e4) ....
     ar  1a16        LQA   ;   increment:= if discmode then 2 else 1;
     gr  10e4  , sc  9e4   ;   output track:= output track - increment;
     grn 1e4   , hhn e29   ;   used tracks:= 0; goto next segment;

c4:  qq  1023  , vy        ; instruction for initalization of by select;
c5:  qq  1e22.39           ; tracks in GP. Only used by drumversion

c6:  pa  2e4   , arn 2e4   ; end comp: return track to  byte out;
     tk  -14   , hs  e11   ;   goto byteout[relative return]
     lk  e3    , vk (e17)  ;
     it (15e4) , hvn e3    ;

     qq [fill]             ;
     qq [GPsum if drum]    ;

d e24=e24+e41, e10=40e10   ; Set GP load parameters
d e22=1e22                 ;
e10: [check load addr]     ;

e                          ; End init comp
```

[30.10.67]
[Gier Algol 4, GP, page 13]

[Init GP: 3 versions, entered with GP in core]

```
< -e44+1                          ; Drum version
d a20=e42                         ;   Define init GP. No code.

×                                 ; Buffer version

a20: arn 14e4  , gt  b27    ;   Set parameters in next seg 2:
     ck  -10   , gt  b28    ;   Set il [unit], il [check]
     arn 11e4  , gr  b30    ;   Set block incr
     arn 13e4  , gr  b32    ;   Set block length
     arn 12e4  , gr  b31    ;   Set current block;
     tk  24    , ck  16     ;   topbuf:= bits(24, 39, current block) + blocklength;
     ar  b32   , gr  b33    ;   GP base:= top buf + GP seg sz pos 19;
     ac  b34   , ar  a22    ;   check transport word:=
     gr  11e4  , ar  a23    ;     check transport word + topbuf;
     us  0     , srn a24    ;   GP to buf ( top buf );
     ar  b32   , gr  12e4   ;   for rest in buf:= - GP size,
     hv  r-1         LT     ;     GP size + block length while rest in buf < 0 do;
     gr  12e4  , hh  a21    ;   goto Init comp buffer version;

a22: qq        t   e41      ;   GP seg sz
a23: qq  e28   t e28-e10-e41;   param word for GP to buf;
a24: qq 40e10.39-e28.39     ;   GP size;

i=39e10                           ;
     qq [GPsum if buffer]         ;

d e24=e24+e41, e10=40e10          ; Set GP Load parameters, fill up this track
d e22=1e22, i=e10                 ;

>                                 ; End buffer version

i=e28                                        ;
     qq e10-e28, hv  a20                     ;   load 2 final words of GP
     qqf a8.9+e4.19+e27.20+e44.21,           ;   a8 = relative address of first segment.

e [ final end GP ]                ;
```

[26.8.67]
[GIER Algol 4, RS, page 1]

[This segment consists of: End pass 8, Initrun, RS code, RS error tracks.

RS error is loaded as a seperate drumblock inside the drumblock holding Endpass 8, Init run, and RS code.
End pass 8 and Init run are loaded with internal references r-marked. RS code is absolute addressed and must have the last instruction in e37 - 1.
The start address for this drumblock during loading is therefore computed below as: e36 = e37 - 40 × number of tracks in RS code - 40 × number of tracks in Initrun - 40 × number of tracks in Endpass 8;
When the segment is taken to core after pass 8 it will include the tracks for RS error with the last instruction in e37-1 and the first in e36 - 40 × number of tracks in RS error]

```
e39=190          ; define size of RS code:
< e27            ;   e39:= basic size of RS code
e39=e39+11       ;   if buffer mode then e39 :=e39+11;
>                ;

<e40+1 , <-e40+1;   if number of units ≠ 0 then
×                ;
e39 = e39 + 51   ;   e39 := e39 + 51;
>                ;
<e40             ;   if number of units > 0 then
e39 = e39 + e40  ;   e39 := e39 + number of units;
>                ;


e36=e37-200      ; define core start for RS:
< e39-200        ;   e36 = first free after RS during run - 5 × 40;
e36 = e36-40     ;   if 5 × 40 < e39 then e36:= e36-40;
>                ;
< e39-240        ;   if 6 × 40 < e39 then e36:= e36-40;
e36 = e36-40     ;
>                ;


e36=e36-40       ; reserve core for init run
e36=e36-40       ; reserve core for end pass 8

b k=e22+e14, i=e36, a50, b50, d40 ; drumblock head for RS;
                                  ;
d33 = e40 + e40 + e40 + e40       ;
d33 = d33 + d33 + 1       ; reserve buffer cells for SAMBA file tables

d1=e38           ; define first track place
d3=4             ;   minimum number of places
d4=164           ;   number of track places from start × length of trackplace
d8=19            ;   max number of trackplaces
```

[END PASS 8]

```
a42: nt (13e4) , qq (rb34) ; Modify init run:
     arn 1e4    , ck -10    ;   stdprtr:= -run track;
     sc  13e4               ;   run track := run track - used tracks;
     arn 3e4    , gt  rb35  ;   start rel:= part 2 (inf 2);
     arn (13e4) D           ;   first tr:= run track;
     ga  rb32   , ck  10    ;   comment also to initial jump in b35;
     ac  rb35   , ps (2e4)  ;
     nt  s3     , pa  rb33  ;   displsz:= -rel stackref 0 - 3;
     ps  s2     , it  sd9   ;   last used:= addr (displ [0]) + 2 + rel stack ref 0;
     pt  rb33   , pi (16e4) ;

     srn d28    D           ; Add RS tracks to compiled code:
     ac  13e4   , tl  -30   ;   run track:= run track - tracks in RS;
     ar  9e4    , gr  9e4   ;   output track:= output track - tracks in RS;
     sr  5e4    , pp r40a42 ;   if output track < first track then
     hs  e5             LT  ;     mess(<<program too big>, 2, 0)
     ac  5e4    , qqn e33   ;   else first track:= output track;

a43: arn 9e4    , hs  e11   ;   for p:= first RS place step 40 until last RS place do
     sk  p      , pp  p40   ;     begin to drum (output track, p);
     arn rd21   , ac  9e4   ;     output track:= output track + 1;
     ncn p-e37+1,hv  ra43   ;     end;

     hv  ra44           NPA ; Pass information:
     hs  e9                 ;   if pass information wanted then
     pmn 1e4    , hs  e8    ;     begin
     srn(13e4) D            ;     new line; print 1 (used tracks, -run track);
     hs  e7                 ;     comment pass 8 used and total used;
     pmn 5e4    , dl  rd20  ;     print (first track : 960);
     ck  -10    , hs  e7    ;     print (first track mod 960);
     cln -10    , hs  e7    ;     end;

a44: pa  2e4    , arn 2e4   ; Exit from translation:
     tk  -14    , hs  e11   ;   to core (exit track , byteout );
     lk  e3     , vk (e17)  ;
     srn(13e4) D            ;   R:= total size.23 + first track.39;
     ck  -14    , ar  5e4   ;
     it (15e4)  , hv  e3    ;   goto byteout [rel exit];

     qq                     ;   fill;
     qq                     ;     -
     qq                     ;     -
     qq                     ;     -
     qq                     ;     -
     qq                     ;     -

d26 = k - e14               ;   first relative RS track;
```

```
[INIT RUN track: May be read into core anywhere between 81e38 (e38=first
 track place) and lower limit for sr0 (which probably is at least 512).
 It is entered in relative 0 which in core must be preceded by four or,
 if buffer mode, six parameters:
 -6: lower limit in buffer        .39; only used in buffer mode
 -5: initial last used in buffer  .39;   -   -   -   -   -
 -4: used in top of free          .39;
 -3: abs track for look up        .9 ;
 -2: abs return track             .39;
 -1: abs track for init run       .39;
by specifies the units to be selected in case of error during run.
Note: Parameters which are set on this track by end pass 8 are marked x]

a40: pp  a40    , hs  ra41  ; Read RS code tracks:
     pp  p40    , ncn p-e37 ;   for p:= first RS place step 40 until last RS place do
[ 2] lk  p      , hh  ra40  ;     to core (next track, p);
     it (rb37)  , pt  d6    ;   next track; set error group in RS code;
     pp (rb36)  , arn d25   ;   set error track in RS code;
     ac  ra40-1 , hs  ra41  ;   param [-1]:= param [-1] + no of error track - 1;
     gp  d6     , pp (rb36) ;   next track; p:= abs group;
b32: qq [xfirsttr], hs a34 ;   set track select parameters and init display;
b33: qq [xdisplsz], ps[xlast used]; comment hsa34 has 4 parameters;
                                  ; Raddr = absfirst track, p = abs group,
a35: pp  p-1    , ar   d23 ;        (s) = first track, (s1) = displsz - 2.
     gr  d2     t -1   MA  ;        Will return to s1h with p = tracktablesize
     bs  p-1    , hv  ra35 ;        and R = track table [1];
<e27,<-e40+1 [buffer mode,no tape units] ; Init track table:
     arn ra40-6, gr  d13   ;   for i:= 2 step 1 until track table size do
x<e27        [buffer mode, tape units ]
     arn b44    , gr d13   ;
>
<e27         [buffer mode]
     ac  d14    , arn ra40-5;    track table[i] MA:= R:= R + place size.19;
     tk  15     , gr  b20  ; if buffermode^-,tape units then lower buf:=R:=param[-6];
x   [end buffer mode]      ; if buffermode^ tape units then lower buf:=R:=cell[b44];
                           ; if buffer mode then begin bufword:=
                           ;   bufword+R; last used in buf:=param[-5] end;
     qq                    ;   fill
     qq                    ;   fill
     qq                    ;   fill
>                          ;
     gs  b20    , pmn ra40-2;
     arn ra40-4, gr  d32   ;   set used in top of free;
     dln rd20   , ck  20   ; Set last used in core and
     ac  d17    , cln -10  ;   exit group and
     ga  d18    , pmn rd24 ;   exit track;
     arn ra40-3 , ga d31   ; Set look up track;
[-2] gm  0              MA ; Set spill exit in abs core 0;
     pm  rd22   , ud  ra36 ;   error unit:= by;
     gk  r-2    , it  (r-2);
     pt  d19    , ud  ra36 ; Set track place word in
     gm  d1               MC ;   first 4 track places;
a36: gm  d1     t e26   MA ;
     it (rb32)  , pt  d15  ; Set first track;
b34: it [xstprtr], pt  d16 ;        ; Set first std proc track;
b35: hsa8,qq[xstart rel+xstarttr.29]; Jump to start compiled program;
a41: arn rd21   , ac  ra40-1; next track:
     pm  ra40-1, dln rd20  ;   M:= param [-1]:= param [-1] + 1;
     ck  -10    , ga  rb36 ;   abs group:= M : 960 + 960;
     cln -10    , ga  rb37 ;   abs track:= Raddr:= M mod 960;
b36: vk[absgr] t  960      ;   select (abs group); select (abs track);
b37: vk[abstr] , hr  s1    ;   return;
```

[26.8.67]
[GIER Algol 4,RS, page 4]
[Note: parameters set by init run are marked ×]

```
d20: qq          960.39    ; constants: 960
d21: qq            1.39    ;   1
d22: qq 512    , hs  a10   ;   track place word
d24 = 39a40 [check load]   ;
d24: qq        , hh  d31   ;   spill exit

     pa  8     , hsf 3     ;
     qq  [fill]            ;   fill track with zeros;
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
     qq  [fill]
i=e37-e39+1                ;   reset i so that RS will be full tracks;

a34: gs  b31   , ac  b30   ; set track select parameters:
     ps  (s)   , pm  a33   ;   comment p = abs group. Raddr = abs track.
     it  s-576 , bs  (b30) ;    (s) = first track. The quantities base and group
     ps  s960  , pp  p1    ;    are computed by the the algoritm below
     gp  d30   , nt  s-512 ;    and RS is initialized as follows:
b30: ps  512   , gs  b28   ;    b1h: a4: group. b27: group - 1.
     nt  s512  , pt  b26   ;    b26h: base - 512. b28: -base;
     it  p     , pt  b1    ;   base:= first track - abstrack;
     it  p-1   , pa  b27   ;   if abs track + 64 > first track then
     gm  d9           MA   ;    begin group:= group + 1; base:= base + 960 end;
```

[26.8.67]
[GIER Algol 4, RS, page 5]
[Note: parameters set by init run are marked x]

```
< e40+1, < -e40+1          ;   if number of tape units ≠ 0 then
x                          ;   begin
e60 = i                    ;
      arn b13   , sr  b29  ; INITIALIZE BUFFER:
      hv  r3          LT   ;   comment all words are initialized
      gr  b13   , ar  b38  ;    to zero with marks 00;
      us  0     , hv  r-3  ;
      arn b45              ;   insert unit numbers in the locations
      arb 29    , it  1    ;   in the buffer which correspond to
      it (b23)  , bs e40+1 ;   the address part of 3c69 in the filetables;
      us 0      , hv r-2   ;
e60 = -e60 + i             ;   SAMBA check;
>                          ;   end;
b31: ps [save s], arn d2   ; initialize display:
a31: gm  d9    t   -1  MA  ;   comment display size - 2 is in (s1);
     pm  a32   , it  -1    ;   display [0]:= display [-1]:= display word 1;
     bt (s1)   , hv  a31   ;   for i:= -2 step -1 until -display size do
     pp  d8    , hh  s1    ;     display [i] := display word 2;
                           ;   return to s1h;
                           ;   comment R = track table[1]; p = track table size;

a32: arn 0     , ga  r1    ; display word 2
a33: qq  0     , hh  a8    ; display word 1

[The code above is part of init run and is overwritten by the track table.
 Core picture when block 0 is entered:
 0:  qq        , hh  c64   ; alarm for spill.
 --
 d1: qqf 512   , hs  a10   ; 1. track place.
 --
 d1+e26:
     qq  512   , hs  a10   ; 2. track place.
 --
d1+e26+e26:
     qq  512   , hs  a10   ; 3. track place.
--
d1+e26+e26+e26:
     qq  512   , hs  a10   ; 4. track place.
--
     arn 0     , ga  r1    ; display [-maxblockno - 1|
     arn 0     , ga  r1    ; ..
     ..                    ; ..
     qq  0     , hh  a8    ; display[-1|
 c:d9:qq 0     , hh  a8    ; display [0|
 d2-d8+1:
     ca  0     , hh  d1+(d8-1) x e26; track table [last place|
     ..
     ..
 d2: ca  0     , hh  d1              ; track table: [1|

         RS-code                     ;

 The track table is further used for run error indication:
   error(n) will set part 2 (track table [d2-r|) to 0 for
   use by the error track]

c49:
d2:  ca  0     , hh  d1    ;  base track table

d9=d2-d8, c =d9            ;  define display [0]
```

```
     [RS code]                ; Is entered here when track not found in table:

     hh  b28            NRC;   if get place then goto search min priority;
a1: c23:                    ; get track: s:= track; vkgr (group);
b1:  ps[xtrack], vk[xgroup];   if s-512 < base - 512 then
b26: it s-512,bs[xbase-512];     begin vkgr (group - 1); vk (s - base + 960) end
b27: vk[xgroup-1], is s960 ;   else vk (s - base);
b28: vks[x-base],ps (b2)   ;   comment base = trackno corresp. to track 0 in group;
                           ; search min priority: s:= i:= top place;
a2:  gs  r1    , arn s     ;   for i:= i - placelength while
     sr  -1    t   -e26    ;     -, mark B[i + placelength] do
     ps (r-1)         NT   ;     if priorpart [i] ≤ priorpart [s] then
     hh  a2           NB   ;       i:= s;

     arn d10   , ac  b14   ; read track: if -,get place then
     hh  r1           NRC  ;     begin tracks transferred:= tracks transferred+1;
     lk  s1    , gs  b11   ;     from drum (track, s+1) end;
                           ;   curr place:= s;
a3:  bs s-d1+512e26t512e26 ; track to table:
     ps  s-e26-1,hv  a3    ;   while s > first place ∨ s < first place -
     it (b1)   , pa s+d2-d1;     place length do s:= s - place length - 1;
     ps (b11)  , arn(b7)   ;   track part [s - first place + base table]:=
                           ;     track; s:= curr place;
                           ; try to reserve new place:
     ca  0     , hv  a4    ;   if track part [table top] ≠ 0 ∧ last used
     is (b20), it s-e26-512;     -512 - place length ≥ top program then
     bs (b3)   , hv  a4    ;     begin table top:= table top - 1;
     gan(b7)   t   -1      ;     track part [table top]:= 0;
     arn b10   , ar  a5    ;     top place:= top place + place length;
b2:  gr d1+d4-e26 t e26 MA ;     store [top place ] mark A:= pack 4
     [top place]           ;       (priority - 4, track is in, 0, op(hs));
b3:  qq d1+d4-512 t e26    ;     top program:= top program + place length
     [top program]         ;     end;

a4:  hh  a10           NRC ;   if -,get place then wait drum;
d30: vk[xgroup], hh  a10   ;   goto set exit addr;

a5:  qq -4.9+123.26+42.35, qq d7 ;   constant: -(arn4D 1), hs a10
                           ;
a6:  ga  b4    , it  0     ; clean up:
a7:                        ;   for i:= top place, i - place length while
b4:  arn -1    Vt -e26 NB  ;     -, markB [i + placelength] do
     hv (a11)  D   -1004   ;     prior part [i]:=
     ar  21    D           ;       if prior part [i] < 491 then -512
     arn 512   D       NO  ;       else prior part [i] - 1003;
     ga (b4)   , hv  a7    ;   priority:= priority - 1004;
                           ;   go to set priority;
```

```
     [next param track: R:= hsc3, qq <rel> + <line no>.29]
c3:  pm  a16   , ud  a16   ;   RC:= 01; track return:= stack param - 1; skip next;
     [jump to next track:  hs c1, qq <rel> + <line no>.29 (f)]
c1:  gr  b21   , arn s     ;   save R:= R;
     gt  b9          IRC ;   rel addr:= set RC (relpart(R));
     arn(b1)   D   1    ;   Raddr:= track:= track + 1;
d19h:hv (b7) , vy[xerrunit];   goto table search [table top]
      [err unit is only used by error tracks]

     [call std proc:   hs c4, <drum point>]
c4:  it (b1)   , pt  b8    ;   return track:= track;
     it (b11)  , pa  b5    ;   return place:= curr place;
b5:c32: qq[return place],gsb6 ;   save s:= s;

     [goto local:      ps<op>, hv c2      ]
     [goto track:      hs c2 , <program point>]
c2:                        ;   save R:= R;
a8:  gr  b21   , arn s     ; goto track no save: R:= param [s];
c26:
a9:  gt  b9          IRC ; point in R: reladdr:= set RC (part 2 (R));

     ck  -10   , ga  b1    ;   Raddr:= track:= part 4 (R);
                           ;   goto table search [table top];
                           ;   comment table search will end as follows:
                           ;   When track already in core: At track is in
                           ;     with track place in s;
b7:                        ;   When track taken from drum: At set exit
c5:  hv  d2-3  , arn b5    ;     addr with track place in curr place
     [table top]           ;     and s;

     [return from std proc: hh c5]
b6:[1c5]ps[saves] , ga b11;   curr place:= return place; RC:= 10;
c62:   [return track]      ;   track:= return track;
b8:  pa  b1    V  -1   IRC ;   go to set priority;

a10: d7=a10-2[used by a5]  ;
b9:  gs  b11   , ps  s -1  ; track is in: curr place:= s;
a11:        [rel addr]     ; set exit addr: s:= s + rel addr;
b10: arn-492[priority] D 1 ; set priority: priority:= Raddr:= priority + 1;
c65:                       ;
b11: ga -1[curr place] VNO ;   if priority = 512 then
     arn b2       hv  a6   ;     begin Raddr:= top place; goto clean up end;
     arn b21               ;   R:= save R;
     hh  s1          LRC ;   if go then go to if LRB then right s1
     hv  s1          LRA ;                        else left s1;
b12: hv[track return] t 1  ;   track return:= track return + 1;
                           ;   goto store [track return];

     [get rel track: qq <track no relative to track>, hs c6]
c6:  gr  b21   , gs  b12   ;   save R:= R; track return:= s;
     arn a16   , it  (s)   ;   track:= Raddr:= track + part 1 (store[s]);
     arn(b1)   D     IRC ;   go:= LRA:= f; rel addr:= 0;
     pt  b9    , hv  (b7)  ;   goto table search [table top];

     [get place:  arn c42, hs c63]
c63: gs  b12   , hv  a9    ;   track return := s; goto point in R;
                           ;    comment marks = 00;
```

```
     [prepare block or call: qq <appetite> , hs c7]
c7:  it (s)     , pa  b24   ;   appetite:= part 1 (store [s]);
     arn b20    , ga  b16   ;   stack addr:= last used; mark A:= f;
                            ;   last used:= last used - appetite;
a15: nt (b24)   , is (b20)  ; test last used:
     it  s-512  , bs (b3)   ;   if last used - 512 ≥ top program then
a16: paf b12    V d5    IRC ; start paramtr:
a17: gs  b17    , hv  a27   ;     begin param addr:= s; goto Next end;
d d5=a17-1                  ;   track return:= start paramtr - 1;  RC:= 01;
                            ; release place:
a18: qq (b3)    t   -e26    ;   top program:= top program - place length;
     qq (b2)    t   -e26    ;   top place:= top place - place length;
     it (b7)    t   1       ;   table top:= table top + 1;
     bs d2-d3+2 , hv  a19   ;   if table top ≤ max table top then
     pt  d2-1   , hv  d6    ;     goto testcore; error (1);

     [expression as formal: ud <formal> , hs c8]
c8:  arn b1     , ck  10    ;   last used:= last used - 1
     gr (b20)   t   -1  MA  ;   stack [last used] mark A:=
     gp (b20)   , nt  s     ;     pack 4 (sr, s - curr place, vk instr, track);
     nt (b11)   , pt (b20)  ;   mark A:= t;

a19: is (b20)   , it  s-512 ; test core: if last used - 512 < top program then
     bs (b3)    , hv  a18   ;   goto release place;
     hh  a22            NA  ;   if -, go then goto test param track;
                            ;   s:= stack addr of formal;
a20: ps (s)     , arn  s    ; end call: R:= stack [s]; p:= sr part (R);
     pp (s)     , hh  p     ;   goto update [p];

     [goto computed: UA:=if undef then 0 else absaddrlabel; ncn (c30) , hs c13]
c13: pp (b22)   , ps  a12   ;   p:=UA; set return( exit to label);

     [exit block: hs c9]
<e27 [buffer version]       ;
c9:  pp (p)     , arn p1    ;   p:= sr part [store[p]];
     mb  d12    , gr  b20   ;   last used word:= store[p+1] ∧ 26 m -
     sr  p1     , tk  15    ;     bits(26,39,store[p+1]) x 2↑15;
     ac  b20    , hv  s1    ;   go back;
x    [core version]
c9:  pp (p)     , pm  p1    ;   p:= sr part [store[p]];
     gm  b20    , hv  s1    ;   last used word:= store[p+1]; go back;
>                           ;

     [mult:   mln <op> X  IZA
              hs  c14  X  NZA]
c14: ar  d10    X       IZA ;   R:= R + 1; swap;
     ar  512    DV      IZA ;   if M ≠ 0 then error (2) else
     pt  d2-2   , hv  d6    ;     R:= R + bit 0; go back;

a22h:hr  s1     , it  s-512 ; test param track: if top program > s - 512 then
     bs (b3)    , hv  a17   ;   param track ok: goto start param tr;
     nt  s      , nt (b11)  ;   curr place:= curr place - s;
     pt  b9     , hv  a1    ;   rel addr:= -curr place; goto get track;
```

[The actions on this page are all part of the parameter transformation mechanism. Each action is headed by the relevant parameter format]

```
     [block information: hv c11, hh displ - <block no> - 1]
c11: gr (b16)  t   -2  MA  ;   stack addr:= stack addr - 2;
     gp (b16)  , pp (b16)  ;   stack [stack addr] mark A:= R;
     gt  r     ; is  -1    ;   sr part [stack addr]:= p; p:= stack addr;
     gp  s1    ; pm  b20   ;   sr part [displ - block no]:= p;
     gm  p1    , hv  a27   ;   stack [stack addr + 1]:= last used cell;
                           ;   goto Next;

     [array: ps (<displ ref>), hv  c12]
c12: arn s     D           ;   stack [stack addr]:= stack [stack addr] + s;
     qq (b24) t1           ;
     ac (b16)  , hv  a27   ;   goto Next;

d11: qq 1.26+1.36          ;   mask, used by 8a27
b25: pa  b22   t [abs addr];   param constant, used by 9a27

     [end call declared or formal: is (<displ ref>), ps s<block rel> f
      other formal or
      procedure identifier:;       is (<displ ref>), pmns<block rel> f]
a23: hh  a20           NZ  ; described in stack: if R = 0 then goto end call;
     hv  a26   X       IRC ;   set RC (swap); goto Store;

     [program point: <2 half words, address part ≥ 0>]
a24: ar  p     DV          ; program point; R:= R + sr; goto Store;
a25: [constant: <40 bits> no marks]
c57:
b24: qq [appetite] t -1    ; constant: appetite:= appetite - 1;
a26: c10:                  ;
b16: gr [stack addr]t -1MRC; Store: stack addr:= stack addr - 1;
a27: c38:                  ;   Stack [stack addr] mark RC:= R;
b17: pmn[param addr]X t1IRC; Next: param addr:= param addr + 1;
                           ;   R:= set RC (store[param addr]);
     hv  a25           NC  ;   if mark 00 then goto constant;
     hv  a24           NT  ;   if R ≥ 0 then goto program point;
     gr  b18               ;   param:= R;
c51:
b18: qq [parameter] ,      ;   execute parameter as two instructions;
     pt  b25               ;   part 2 [param const]:= abs addr;
     hv  a23           LRC ;   if RC then goto described in stack;
     [absolute addressed:
      simple: ps(n) (<displ rel>) (,) it (n) s<blockrel> (f)
      literal:ps n (stack addr)   (,) it (n) s<stack addr rel> (f)]
     arn b18   , mb  d11   ; Abs addr: R:= param const +
     ar  b25           IRA ;     bit (26, parameter) + bit (36, parameter);
d15: hv  a26, ps[xfirsttr] ;   RA:= 1; goto Store
     [first track only used by error track]
```

```
     [case i of : Raddr:= value of i;
                  qq <no of params> , hs c15]
c15: ar (s)    D        LO  ;   index:= Raddr;
     ga  b19   V        LT  ;   if index > no of params ∨ index ≤ 0 then
     hv  s1                 ;     go back; comment to alarm exit;
b19:                        ;
c16: arn s[index] t 3 IRC   ;   R:= UV:= case param:= set RC (param [index]);
     gr  b23   , gr  b15    ;   if NC then goback 1;
     hv  s2             NC  ;     comment constant;
c52:
b15: qq [execute param] ,   ;   execute case param;
     pa  b22           NRA ;   if label then UA:= abs addr of label;
     gr  b23   V       LRB ;   if float case then
     nkf 39    , grf b23   ;     UV:= RF:= float (value(simple))
d16: hv  s2, ps [xstdprtr] ;   else UV:= value (simple); go back 1
     [stdprtr only used by error track]

     [formal addr: ud<formal> , hs  c28
                   qq w       , arn b22]
c28: arn(s1)                ;   R:= description word;
     hv  s2             NZ  ;   if R ≠ 0 then go back 1;
c25: pt  d2-9  , hv  d6     ; assign error: error (9);

     [end subscr in buffer: UV:= value; R:= description, hv c18]
c18: [end UV, R, RF - expression: UV:= value, hvn c18]
a28: pa  b22   t   b23      ;   UA:= address (UV);

     [end subscr in core: UA:= R:= address of value, hv c19]
     [end addr expr: UA:= address of value, hvn c19]
c19:                        ;   s:= last used;
a29h:ps (b20)  , gr  b21    ; expr return: save R:= R;
     it  s1    , pa  b20    ;   last used:=s+1; p:=sr part[s];
     pp (s)    V            ;   goto update and go;
a12 = i - 1                 ; exit to label:
     ps (b22)               ;   s:= UA;
     arn s     , hh  p      ;   R:=stack[s]; goto update[p];

     [next in: ud c37, hs c37]
c37: lyn p 0   D            ;

     [next out: qq <char to output> , hs c39]
c39: sy (s)    , hv  s1     ;
     [c37 and c39 are used for all input and output and may be
      replaced by other instructions having the wanted effect]
```

```
[The actions below are loaded in one of two versions: one for
 arrays in buffer and another for arrays in core]

< e27 [buffer versions]

     [reserve array: arn length , hs c20]
c20: ac  p1    , pm  d12   ;   stack[sr+1]:= stack[sr+1] + R;
     cm  0     D       NZ  ;   if R <0 ∨ R > 2↑14 - 1 then
a30: pt  d2-3  ; hv  d6    ;     error(3);
     ck  15    ; sc  b20   ;   last used word:= R := last used word -
     arn b20   ; tk  11    ;    R x 2↑15; R:= bits with first as sign
     tk  -26   , sr  d13   ;    (10,25;R) - lover buf limit;
     hv  a30         LT    ;   if R<0 then error(3);
     ar  d14   , hv  s1    ;   R:=R + buf array word; go back;

d12: qq  -1.25             ; mask , used by 1c9 , c20 , and 1c22
c36:
d13: qq 0[x+lower buf.39]  ; set by init run
d14: qqb23t1[x+lowerbuf.39]; set by init run

     [exit func: ps p + number of formals + 2, hv c21]
c21: pm  p-1   , ud  a28   ;   UV:= stack [p-1]; UA:= addr(UV);

     [exit proc: ps p + number of formals + 2, hh c22]
c22h:gm  b23   ; arn p1    ;   last used word:=''
     mb  d12   , gr  b20   ;     stack[sr+1] ∧ 26 m; R:=0;
     hhn a29               ;   goto expr return;

     [assign to formal subscripted: ud formal ; hs c24
                                    qq  w     , store op (b23)]
c24: pa  b22   t  b23      ;   UA:= addr (UV);
     gr  b21   , ud  s1    ;   save R:= R; execute (store op(UA));
     arn(s1)               ;   R:= description of buffer word;
     us  0     V       NZ  ;   if R = 0 then
     pt  d2-9  ; hv  d6    ;     error (9)
     arn b21   , hv  s2    ;   else us 0; R:= save R; goback 1;

x    [core versions]

     [reserve array: srn length, hs c20;
                     arn last used, hv s2]
c20: arn b20   V       LT  ;   if R < 0 then error (3);
a30: pt  d2-3  ; hv  d6    ;
     ck  10    , sr (s)    ;   R:= (last used shift 10) - length;
     hv  a30         LT    ;   if R < 0 then error (3);
     ck  -10   ; ga  b20   ;   last used:= stack [p+1] := R shift - 10;
     ga  p1    , hh  c7    ;   goto test last used;
                           ;     comment ensure NA, therefore jump to c7;
```

```
        [core versions ...]

     [exit func: ps p + number of formals + 2, hv c21]
c21: pm  p-1   , ud  a28   ;   UV:= stack [p-1]; UA:= addr(UV);

     [exit proc: ps p + number of formals + 2, hh c22]
c22h:gm  b23   , hhn a29   ;   R := 0;  go to expr return;


     [assign to formal subscripted: ud formal , hs  c24
                                    qq  w       , store op (b23)]
c24: gr  b21   , arn(s1)   ;   save R:= R; R:= description of actual;
     ga  b22   V       NZ  ;   if R = 0 then
     pt  d2-9  , hv  d6     ;     error (9)
     arn b21   , hh  s1     ;   R:= save R; go right back;

>    [end core versions]

d32:
c31: qq[x used in top of free];
d31:
c64: qq[xcatalog], pt d2-10; d31h: spill entry:  error (10);
c27: [error: pt<base table - error number, hv/hs c27]
d6:  vk[xerrgr],vk[xerrtr] ; error: to core (error track, first place);
     lk (d17)  , ud  d17   ;   goto first place;
c29: [exit: hhn c29]
d17: hv1d1,vk960[xexitgr]  ; exit: to core (exit track, first place);
d18: vk [xexittr], hv 1d6  ;   goto first place;


     [SAMBA-sequenses]

<e40+1 , <-e40+1           ;   if number of units ≠ 0 then
x                          ;   begin
     [get filetable : R := unit.9; hs c66]
c66: ga r2      , tk -27   ;   paramword := 3xunit+ c56+8.19-7.39;
     ar d35     , gr  b47  ;
c67: bt[unit]   t -e40+511 ; if unit<1 V unit > number of units
     il  0      , hr  s1   ;   then error(<param>);
     pt  d2-17  , hs  d6   ;   transfer filetable; return;
```

```
     [transfer and calculate sum: R :=paramword1; M := paramword2;
                                                    hs c48; ]

c48: xr                    LOC ;   if arrayΛ outrec then change RM;
     il          X         LTB ;   if arrayV inrec  then transfer in and change RM;
     us          X         LTA ;   if arrayV outrec then transferout and change RM;
     hv  s1                NPB ;   if -, sumcheck then return;
     xr                        ;   change RM;
     ga  b41   , gt  r1        ;   b41 := base transfer addr in core; save return;
     gs  b40   , psn[length];  R := 0; s:= number of words transferred;
     ps s-1    , ar(b41)       ; NEXTWORD: s:= s-1; R := R + core[base +s];
     bs  s     , hv r-1        ;   if s>0 then go to NEXTWORD;
b41: qq s[base], ac  b44       ;   sambasum := sambasum + R;
b40: hv  [save s]t  1          ;   return;


     [exit SAMBAproc  :   hv c40  or hv c46 ]

c40: arn b47   , us  0         ;   restore filetab;
c46: hh  rc5   , ud  c18       ;   if -, in error then go to EXIT ST PROC;
     pa  c46   t  c5 - c46     ;   UA := address(UV); in error := false;
     ps  p+2   , hh  c22       ;   s := stackref + 2; go to EXIT PROC;

     [get statusword :  hs c60]

c60: arn  d34  , is (b45)      ;   R := b46.9+1.19+0.39;
     il  s224  , il  0         ;   get statusword to buffer0(unit);
     arn b46   , hr  s1        ;   R := b46 := buffer[0]; return;

>                              ;   end samba sequenses;

     [constants]

c41: d10:                      ;
c42: qq 1.39                   ;    1
c58: qq 10.39                  ;   10
c43:                           ;
c44: qq -1 t 256               ;     .5
c53: qq  0 t 256               ;    1.0

     [variables]

c35:b20: qq[xlastused in core.9+last used in buffer.25] f ; last used
c55:     vy[select bits] 48     t  [mask] 768             ; select
c61: b14: qq                   ; tracks transferred
c30: b22: qq                   ; UA
c33:      qq                   ; address
c54:      qq                   ; char
```

```
<e40+1 , <-e40+1          ;   if number of units ≠ 0 then
×                         ;   begin
c45: b46: qq              ;   status word location

c82:      qq              ;   sambaerror;
c47: b44: qq d33.39       ;   sambasum; constant used by init run;
c81: b47: qq              ;   paramword for current filetables;
c73:      qq              ;   working cell; constant used by init run;
c74: b38: qq b23  t 8     ;      -     -     -     -   -   -   - ;
c79: b29: qq 8.39         ;      -     -     -     -   -   -   - ;
c80: b13: qq 4096.39      ;      -     -     -     -   -   -   - ;
i = c73 + e69             ;   reserve place for SAMBA record segment ;
>                         ;   end;


c50: b41: qq              ;   save R;
c34: b43: qq              ;   UV-4
c68:      qq              ;   UV-3
c70: d25: qq              ;   UV-2 ; number of errortracks.39.
                          ;   loaded by errortracks. Used by init run;
c72: d23: qq te 26        ;   UV-1 ; placelength.19 .  Used by init run;
c17: b23: qq              ;   UV
c56:      qq              ;   UV + 1 ; filetab[1];

<e40+1 , <-e40+1          ;   if number of units ≠ 0 then
×                         ;   begin

[1c56:]   qq              ;            filetab[2];
[2c56:]   qq              ;            filetab[3];
[3c56:]   qq              ;            filetab[4];
c69:      qq              ;            filetab[5];
[1c69:]   qq              ;            filetab[6];
[2c69:]   qq              ;            filetab[7];
[3c69:]                   ;            filetab[8];
c71: b45: qq  b23  t 1    ;   constant used by init run;
>                         ;   end;


<e40                      ;   if number of units > 0 then
                          ;   begin
          qq              ;   unit 1 error;
          qq              ;   unit 2 error;
          qq              ;   unit 3 error;
          qq              ;   unit 4 error;
          qq              ;   unit 5 error;
          qq              ;   unit 6 error;
          qq              ;   unit 7 error;
          qq              ;   unit 8 error;

i = c71 + e40 + 1         ;   delete superflous error-cells;
>                         ;   end;
```

[26.8.67]
[GIER Algol 4, RS, page 15]

```
<e40+1 , <-e40+1                ;   if number of units ≠ 0 then
×                               ;   begin

     [samba constants]

c75: d34: qq b46   t 1          ;
c76:      qq  1                 ;
c77:      qq       t 1          ;
c78:      qq  1.25-1.39         ;   mask;
d35:      qq  c56.9+8.19-7.39;  filetab base addr;

e60=e60+c60+c40+c46+c71+c72+c56+c71  ;
e60=-e60+c66+c67+c48+c45+c81+c80+c45 ; SAMBA check;

>                               ;   end;


c59=1023                        ;   output sum cell

e37:                            ;   control definition of top RS in core;
e19=c+c1+c7+c11+c13+c20+c24+c17 ; RS check entries
e19=e19+c1+c11+c20+c17           ; system
```

```
[Error tracks. Two tracks which are taken in to trackplace 1 and 2]
d27=k-e14 [define rel first err tr];
b k=d27+e14, i=1d1, c18                 ; Error track 1, track place 1.

[Upon exit 1d1-5d1 will hold the saved registers and 6d1-9d1 err. inf:]

[1d1]gi  1d1          IRC ;  gi <indicator>  IRC
[2]  gs  2d1    , ud  d19 ;  gs <s> , ...
[3]  gr  3d1          IOA ;    < R >
[4]  gp  4d1    ; arn d6  ;  gp <p> , ...
[5]  sy  64     ; gm  5d1 ;    < M >
[6]  gt  6d1    ; ps [errtr]; qq <error no> , qq
[7]  ga  9d1    ; can s-959 ; ga <error track> , ...
[8]  ps  -1     ; it  1   ;    < from line >
[9]  vk [errgr]; vk  s1   ;    < to line >
     lk  42d1   , vk (9d1) ; Start error: 1d1: save registers; select(errunit);
                           ;   writecr; to core(error track 2, track place 2);
c1:  pp  -1     ; arn pd2 ; Get error no: p:= -1;
     ck  10     , nc  0   ;   while part 2(table[table base+p]) ≠ 0 do p:= p+1;
     pp  p-1    ; hh  c1  ;
     arn p1d2   ; ar  r1  ;   comment p now holds - error no  (<0);
     gt  pd2    ; xrn e26 ;   restore part 2 of error table word;
     bs  p11    , nt  p   ; Write error text:
     arn c3     , sy  58  ;
     sy  29     , cl  40  ;   write char(<LC>); write char(<red>);
c2:  cl  -6     ; ck  -4  ;
     ca  63     ; ar  r1  ;   write text(error text [if -p > max err no then
[1]  ga  r1     ; nc  10  ;                                  0 else -p]);
[1]  sy[charac], hvn c2   ;
     srn p      DX        ;   save error no;
     gm  6d1    , bs p-502 ;  if -p > max err no then
     cl  -30    , hsn c17 ;   write integer ({p}, -p);
< e40                      ;   if number of tape units > 0 then
     sy  0      , sy (3c69) ;  incorporate( write integer({dd}, tape unit));
>                          ;
     sy  0      , hv  42d1 ;  write char (0);  goto track place 2;

c3: terror ;[n] error texts:
    tstack;; 1
    tmult; ; 2
< e40+1, < -e40+1          ;   if number of tape units = 0 then
    tarray;; 3             ;    incorporate({<array>})
×                          ;    else
    tbuffer; 3             ;    incorporate({<buffer>});
>
    tcase; ; 4
    tindex;; 5
    tln;   ; 6
    tsqrt; ; 7
    texp;  ; 8
    tformal; 9
    tspill;;10

    qq[fill]
    qq[fill]
< e40                      ;   if number of tape units ≤ 0 then
×
    qq[fill]               ;    incorporate(<fill>);
>
```

```
[26.8.67]
[GIER Algol 4, RS, page 17]

[Error track 2, loaded frm 41d1 but executed from 42d1]

c4=41d1 [check load address]
c4:  arn b8     , pp (b1)    ; Find line numbers:
     qq         , ud  d16    ;
     nt  p-511  , bs  s-512  ;
     gt  r      , pp[ret.tr];   p:= if curr track ≥ std pr tr then
     sy  0      , ud  d15    ;     return track else curr track;
                             ;   write char (0);  s:='first track;
c5:  it  p-576  , bs  s-512  ; loop 1: if s > p-64 then
     ps  p-65   , pa  rc8    ;     begin found:= t; s:= p-1 end
                             ;   else s:= s+64;
c6:  ps  s+64   , ud  a1     ; Get line number on track s:
     ud  1a1    , ud  1a1    ;
     ud  2a1    , ud  2a1    ;   to core (run track(s), track place 3);
     ud  3a1    , lk  83d1   ;
     vk (2a1)   , pmn 122d1  ;   Raddr:= line mod 1024;
     gp  7d1    , cln -10    ;   if Raddr < 512 then
c7:  qq  0      V  0    NO   ;     begin times 1024:= times1024+incr; incr:=0 end
     it  1                   ;   else incr:= 1;
     pt  rc7    , ck -10     ;   error track := p;
     ar (rc7)   DX           ;   M:= to line:= times 1024 x 1024 + Raddr;
     cln-20     , gm  9d1
c8:  bs  1      , hv rc5     ;   if -, found then goto loop 1;
c9:  bs  1      , gm  8d1    ;   if -, found 1 then from line:= M;
     hs  rc12                ;   write integer (⟨p⟩, M);
     can(rc9)   , hv  rc10   ;   if -, found 1 then
     ps  p      , sy   32    ;     begin write char (<->);  s:= p;
     pa  rc9    , hh  rc6    ;       found 1:= t; goto Get line number end;

c10: pmn 7d1    X            ; Finish: write char (<,>); write char (0);
     sy  27     , sy  0      ;
     cl  10     , hs  rc12   ;   out integer (⟨p⟩, error track);
     ps  1d1    , arn 6d1    ;   s:= address of first information ;
     hh (d17)   D   41       ;   Raddr:= error no;
                             ;   exitplace:= exitplace + 41; goto exit;

c11: m  10-3                 ; constants for write integer
[1]     10                   ;

c12: mln rc11   , pa  rc15   ; procedure write integer;
c13: ck  -10    , bs (rc16)  ;
c14: pa  rc15   t  16        ;
c15: arn 0      D      IZ    ;
     ga  r1     , nc  0      ;
[1]  sy  0      , ud  rc14   ;
c16: ncn 0      t  -256      ;
     mln r1c11  , hv  rc13   ;
     hr  s1                  ;

c17=c12+1                    ; define entry for write integer from error track 1
c18=81d1                     ; check load address
c18: qq [RS sum]             ;

e22=k-e14, e47=j             ; set load parameter
d29=e36-80                   ; define RS segment start
e [error tracks]             ;   for two error tracks (80 words)
```

[26.8.67]
[GIER Algol 4, RS, page 18]

[Finish loading of RS segment]

```
d28=e22-d26              ; number of RS tracks  (Init run + RScode + error tracks)
i=d25                    ; load number of error tracks.39 in init run
qq e22.39-d27.39-1.39;

b k=e23, i=0             ; load RS segment word
i= 16e21                 ;
qq d29.9 - 1.9 + e37.19 - d29.19 + 3.20 + d29.39 - 1.39 f ;
e                        ;
e [RS]                   ;

d e49=2                  ; set next tape number;
```

[after i follows STOPCODE, SUMCODE and a sum character]

```
i  T1,L1
s
```

[Here follows STOPCODE and CLEARCODE]

```
<e49-1, <-e49+3, x  ;   test tape number
i wrong tape
s                   ;
>                   ;

d e48=1             ;   translator is in:=true
d e52=9             ;   version number

<e52-e50,           ;   if version number T2 gr max version number
                    ;   then the following definitions are loaded;
d e61=1             ;   define version number T1 and L1
d e62=e52           ;   define version number T2
d e63=3             ;   define version number T3
d e64=4             ;   define version number T4
d e65=5             ;   define version number T5
d e66=6             ;   define version number T6 and L2
d e67=7             ;   define version number T7 and L3
d e68=8             ;   define version number T8 and L4
d e50=e52
>                   ;
```

[10.4.67]
[Gier Algol 4, pass 1, first page]

[CONTENTS OF TAPE:

Special requirements: Between input table lower case [0| and input table upper case [0| must be at least 128 words as this is the range of an input symbol from reader.
Allowed input bytes are in the table flagged by an f-mark.
All input bytes outside tables are forbidden. Consequently no word outside tables from input table lower case [0| to input table lower case [127| and from input table upper case [0| to input table upper case [127| may be f-marked.
end contents .....]

[Drum block head, pass 1]

```
b k=e22+e14, i=e2-e47, a33, b29, c57, d97;
i=e2                                      ;

e2:  arn 81e13 t   1        ; Last part of forward pass track repeated
     ga  e13-4 V            ;   with 1e2 and 2e2 slightly changed.
     qq                     ;
     tk  10    , ga  e13-3  ;
     tk  10    , ga  e13-2  ;
     tk  10    , ga  e13-1  ;
e1:  arn e13-1 t   -4       ;
     hr  s1                 ;
     arn 10e4  , ac (s1)    ;
     arn (s1)  , hv  e11    ; Last 4 words replaced by this one;
```

[1. INDICATOR USAGE, COMMON ADDRESS VARIABLES

[INDICATOR USAGE:
NZA = normal. Initialized to false.
LZB = comm ok. In strings: LOB = set case. Initialized to false.
LTA = Poff active, initialized in next segm 3 from mode bits.
LTB = sum. Initialized to false. Set to true by SUMCODE and CLEAR CODE.
LPA = print. Initialized to true.
LPB = line test. Indicates that each line interval lines should be printed. Initialized in next segm 3 from mode bits.
LQA = lined. Indicates that the present character is underlined.
LQB = comm. Indicates that the present character is underlined or barred.
RA Used internally in code
RB - - - -

[b-NAMES COMMON TO WHOLE PASS 1, SPECIFYING ADDRESS VARIABLES:
Form of description:
Slipname: part no of word, variable name, initial value, comment.

b10: 1, char, - , last character input.
b10: 2, case, d50, address of lower or upper case input table.
b11: 2, action63 , 0, set by copy to: c2(reader), 2c2(drum),or, a21(keys)
b12: 2, print medium, 16e4, during message set to 15e4
b13: 1, outCR, 0, after first begin set to d22, incode to d2 or d39.
b14: 1, case1, 0, set by UC to d53, by LC to 0.
b16: 1, sem value, d12, d1 from code to;
b17: 1, end level, 0, increased by 1 by begin, decreased by end.
b18: 1, colon value, d27, d16 from code to ;
b19: 1, comma value, d16, d27 from code to ;

[b20 - 4b20: Full word variables, see page 4.

b22: 2, exit address from store string, next string word, changed by finis pass 1.
b24: 1, const kind , - , used by output of layouts and strings.
b25: 1, str case , - , current case in strings LC = -1, UC = -3.
b26: 1, i copy, d50, stack pointer for copy.
b27: 1, search, set to abs track for search by init pass 1.
b28: 1, init medium, - , set to abs track for init medium by init pass 1.]

[2. DEFINITIONS OF OUTPUT VALUES AND STRING VALUES REFERRED TO OUTSIDE TABLES.
Kinds: v: normal output; c: in code; s: string; cs: string in code

Kind Meaning]

```
d1  = 1008; v   begin code
d2  =   76; c   blind CR
d3  =  249; v   code
d4  =   74; c   :
d5  =   61; c   -
```

[d6-d11, see tables, page 28]

```
d12 =  167; v   ;
d13 =  172; v   end
d14 = 1009; v   end pass
d15 =  957; v   <lined digit> - <digit>. 0 - 9 is output as 1014 - 1023.
d16 =  220; v   ,
d17 =   14; scs _
d18 =   63; scs CR
d19 =   10; s   }
d20 = 1011; v   short string
d21 = 1012; v   long string
d22 = 1010; v   CR
d23 =  291; v   )
d24 =   93; v   begin
d25 =  287; v   fat comma
d26 = 1013; v   layout
d27 =  194; v   :
d28 =  273; v   =
d29 =   76; v   -
d30 =  228; v   :=
d31 =  280; v   =>
d32 =   82; v   -,
d33 =   98; v   for
d34 =  176; v   else
d35 =  184; v   (
d36 =   32; c   forbidden
d37 =   59; c   t1
d38 =   60; c   t2
d39 =   63; c   CR
d40 =   15; s   word end
d41 =  271; v   <
d42 =   72; v   +
d43 =  275; v   >
d44 =   86; v   goto  go to  go to
d47 =   62; c   f
d48 =   77; c   e
```

d53 = 128; special, UC1, used to test current case.

[3. CONSTANTS, VARIABLES, INIT PASS 1, RESERVATION LOWER CASE TABLE]

```
[CONSTANTS                          Name(s)                    Used in action
                                                               - when more than once]
a30: qq 1.4 + 1.23 + 1.27  ; d increment[0]                    c44
[1]  qq 1.23               ; b bit                             c44
[2]  qq 1.34               ; n bit                             c44
[3]  qq 1.29               ; sign bit                          c44
[4]  qq -1.7+1.23-1.39     ; mask used in copy                 c40
[5]  qq 1e13   t  1.29 + 4 ; il-param for buf medium           c40
[6]  qq 1.4 + 1.23 + 1.33  ; d increment[6]                    c44
[7]  qq 1.39               ; bit 39                            c40,c44,c45,c10
[8]  pmn(e1)   X   1       ; input instr, drum or keys         c40
[9]  ga  b10               ;   -     -  ,   -  -   -           c40
[10] lyn b10   V           ;   -     -  , reader               c40
[11] hv  a3          IT    ;   -     -  ,   -                  c40
[12] qq 1.3 + 1.37         ; d increment[12], bit 37           c44,c45
[13] qq  6.23              ; char pr word                      c40
```

[Variables. Full word variables and working areas overwrite the code for init pass 1 (see below).

	Name(s)	Used by
b20:	word 0, spaceword	c40, c44, c45, c50
1b20:	word 1, layoutword	c44, c45
2b20:	word 2	c45
3b20:	word 3	c45
d49: to 6d49:		
	copy name[0-6]	c40
7d49: to d50-1:		
	copy stack	c40

The copy stack starts at d50-1 and works towars d49. It is bounded by d50 and 6d49 which both are f-marked.]

```
a31:                              ; Init pass 1:  normal:= commok:= sum:=
b20: pi  544   t   144            ;     lined:= comp:= false;  print:= true;
[1]  arn -2    , ga  b27          ;   set search and init medium track numbers;
[2]  tk  10    , ga  b28          ;
[3]  arn  7e4  , hs  c52          ;   new medium(first medium);
a32:                              ;     comment returns with first character read;
d49: nc  2[b]  , hs  c1           ; read first begin: if Raddr ≠ tb then
                                  ; start begin:  next 1;
     qq  5[e]  , hsn a33          ;   begin test (5);
     qq  7[g]  , hsn a33          ;   begin test (7);
     qq  9[i]  , hsn a33          ;   begin test (9);
     qq 14[n]  , hsn a33          ;   begin test (14);
     pa  b13   t   d22 IQC        ;   comp:= lined:= false;  outCR:= vCR;
[7]  pif 0     t   991 NPB        ;   if -,linetest then print:= false;
     pa  14e4  , hv  c28          ;   string rel:= 0;  goto begin;

a33: hs   c1            LQA ; begin test:  if lined then next 1;
     hh  a32            NQA ;   if -,lined then goto start begin;
     nc (s)    , hv  a32    ;   if  Raddr ≠ (test value)  then goto
     arn a33   , hr  s1     ;    read first begin;  Rmarks:= 0;  return;

     qq [first medium is stacked here];

d50:       ; Input table lower case[0].  Limit copy pointer + 1.
i = 65i    ; reserve input table lower case [0:64], loaded later.

d52:       ; Input table lower case[65], secondary table words.
i = 3i     ; reserve secondary tablewords.
```

[4.1. CENTRAL INPUT]

```
c1:  pm  r     V       IQC ; next 1: comp:= lined:= f;
                 , ps  i    ;   comment ps i will ensure return to
                            ;   next as the outermost return level;
```

[Input instructions: The next 3 instructions depend on the current input medium as follows: They are set from copy as follows:

	reader input:	key input:	drum input:	buf medium:
c2:	lyn b10 V	pmn(e1) X 1	pmn(e1) X 1	pmn(e1) X 1
	hs irrelevant LA	hs e2 LA	hs a14 LA	hs a16 LA
	hv a3 LT	ga b10	ga b10	ga b10

note: a14 and a16 are local in c40]

```
c2:  zq [input instr 1]          ; next: Raddr:= char:=
     hs [next word action]  LA   ;       next character from input
     zq [input instr 2]          ;   if reader input ∧ R < 0 then
                                 ;     goto test it

b2:  ac[sum 1] Dt  1  LTB  ;   if sum then sum 1:= sum 1 + Raddr + 1;
b10: pm  -1    X   d50     ;   char:= char + case; M:= R;
     [char]      [case]    ;   R:= input table[char];
     hv  a3    X       NB  ;   if -,markB then begin swap; goto test it end;
     hv  a6    X       LPA ;   if print then begin swap; goto print it end;
                           ; after print:
a2:  hv (b10)          LA  ;   if LA then goto central action [Raddr];
                           ; after central: if inpoff then goto next;
c3:  hr  s1            LZA ;   if -,normal then return;
     [modified by pon.poff]; in normal:
c4:  hs  c50           LQB ;   if comp then compound;
                           ; treat normal:
c5:  ga  b1    V       LT  ;   if R ≥ 0 then
                           ; normal out:
c6:  mb  1023  DV          ;   begin Raddr:= part 1(R); normal:= t;
b1:  hv  -1    t   c   IZA ;   commok:= f; goto output end;
     [normal action]       ; normal act: normal:= t;
     hv  e3            IZC ;   goto normal action (R addr);
c7:  hsn c1            IZA ; comp or simple:
     hv  c50           LQB ;   next 1; if comp then goto compound;
     hr  s1                ;   return;
```

[4.1. cont. and normal action for SPACE]

```
a3:  ck  4      , nc  63.5  ; test it: if bits(4,9,R) ≠ 63 then
c9:  hs  c49                ; forbidden: mess 1(≪character≫, 0, 1);
c19: hv  c2     , qq  sd54  ; space: blind: goto next;

a6:  ga  b4     X           ; print it:
b12: ck  -5     , ud  16e4  ;   begin select(print medium);
           [print medium]   ;   if -,no printing then out char(Raddr);
b4:  sy  -1           NO    ;   select (normal medium);
     arn(b10)   , ud  16e4  ;   swop; goto after print
                            ;   end;
b11: hv a2, hv [action 63]  ; action 63: goto
    [set by start medium    ;   if -, input from keys then next
     and input from keys]   ;   else if during input from keys then new key line
                            ;   else finis;
```

[4.2. CENTRAL ACTIONS]

```
c10:
b13: pmn[outCR]DX           ; CR: if out CR = 0 then goto next;
     hv  c2           LZ    ;     comment pre- or post-lude;
     hs  e3                 ;   output(outCR);
     arn 7a30  , ac  e4     ;   CRcount:= CRcount + 1, print:= f;
     pm  e4           IPA   ;   if linetest ∧
     dln 3e4   X      LPB   ;       CRcount mod line interval = 0 then
     hv  a7           NZ    ;         begin
     sy  58    , pmn e4     ;         outchar (58); print:= t;
     hs  e9                 ;         newline;
     hs  e8                 ;         print1(CRcount); comment LA on return;
b14: bs  0     , sy  60     ;         if case 1 > 0 then
       [case 1]             ;           outchar(60);
     sy  0            IPA   ;         outchar(0);
                            ;         end;
a7:  pm d18     DV     LZA   ;   if -, normal then M:= s CR
     hv  c2           NQB   ;   else if -, comp then goto next;
     arn 2d52  , hv  c3     ;   R:= table [second CR word]; goto after central;
```

[4.2. cont.]

```
c11: it  d53    , pa  b14    ; UC: case:= baseUC; case1:= UC1; goto next;
     gt  b10    , hv  c2     ; LC: case:= baseLC; case1:= LC1; goto next;

c12: arn d52   V        LQB ; Line:
     hv  c2             IQC ;   if comp then R:= table[second lineword]
     hv  c3             NQA ;   else begin comp:= lined:= t; goto next end;
a8:  hv  c2[used by poff]  ;   goto if lined then next else after central;
[+1] hr s1[used by pon]LZA ;

c13: arn 1d52  V        LQA ; bar: if -, lined then
     hv  c2             IQB ;   begin comp:= t; goto next end;
     hv  c3                 ;   R:= table[second bar word];
                            ;   goto after central;

c14: hv  c2             NTA ; poff:  if -, Poff active then goto next;
     pm  a8     , gm c3     ;
     hs  c49                ;   action after central:= goto next
     hv  c2     , qq  pd55  ;   mess 1 ({<off}, 0, 3); goto next;

c15: hv  c2             NTA ; pon:  if -,Poff active then goto next;
     pm  1a8    , gm  c3    ;   action after central:= conditional return;
     hs  c49           IQC ;   mess1 ({<on}, 0, 3);
     hv  c2     , qq  pd56  ;   comp:= lined:= false;  goto next;


c16: arn(b2)   D t  -62     ; Sum code:  sum:= sum - 62; comment the sum code;
     ck  -5    , ar  b2     ;   sumcheck:= (sum : 32 + sum) mod 32
     mb  31    D            ;
     ga  b6    , pm  1c2    ;   store second input instruction below;
     gm  r1    , ud  c2     ;   execute first input instruction;
     qq[sec. input instr]   ;   second input instruction;
b6:  nc  -1    t   31       ;    comment next character now in Raddr;
     [sum check]            ;    if Raddr ≠ sum check + 31 then
     hs  c49                ;      mess1({<sum>, 0, 3);
c17: pa  b2    , qq  pd68   ; clear code: sum 1:= 0;
     pi  1.3   t  959       ;   sum:= t;
     hv  c2                 ;   goto next;
```

```
[4.3. NORMAL ACTIONS]          ; comment Note the return to s in
                               ;   read comment;
c18: arn r     , ud  15e4      ; message: print medium:= alarm print;
     gt  b12   , sy  64        ;   select(alarm medium); outchar(CR);
     hs  a9            IPA     ;   print:= t; read comment;
     arn r2    , ud  15e4      ;   select(alarm);
     sy  58    , ud  16e4      ;   outchar(LC); select(normal); print:= f;
[2]  pt  b12   Vt  16e4IPA     ;   printmedium:= normal; goto fin comm;

c20: hs  a9                    ; comment: read comment;
     hr  c1            IZA     ; fin comm: normal:= t; return to next 1;
a9:  hs  c49           NZB     ; read comment: if -, commok then
     hv  r1    , qq  sd57      ;   mess1 (‡<comment›, 0, 1);
     hsn c1            IZC     ;   commok:= t; normal:= f; next 1;
     nc  d8            NQB     ;   if Raddr ≠ t; ∨ comp then
     pmn r     , hr  s         ;     begin set mark A; return same end;
     hr  s1                    ;   return;

c21:
b16: pmn d12[v;] DX    IZB     ; semicolon: commok:= t; Raddr:= sem value;
     [sem value]               ;   if Raddr = vbegcode then goto code 1;
     ca  d1    , hv  c31       ;   normal:= t;
     hv  e3            IZA     ;   goto output;


a10: hsn c1            IZA     ; rep end comm: normal:= f; next 1;

a11: hs  c50           LQB     ; test sem: if comp then compound;
     ca  d8    , hv  c21       ;   if Raddr = t; then goto semicolon;
     nc  d34   , ca  d33       ;   if Raddr = tfor ∨ Raddr = telse then
     hv  c5                    ;     goto treat normal;
     nc  d7    , hv  a10       ;   if Raddr ≠ tend then goto rep end comm;

c22: pmn d13   DX              ; end: comm ok:= f;
     hs  e3            IZB     ;   output(vend);
b17: bt  -1    t   -1          ;   endlevel:= endlevel - 1;
     [endlevel]                ;   if endlevel ≥ 0 then
     hv  a10                   ;     goto rep end comm;

     pmn(14e4) D       IZA     ; finis pass 1: M:= str rel; normal:= f;
     pt  b22   t   i           ;   set return from string track out to here
     pp  0                     ;   ensure return;
     bs (14e4) , hv  c51       ;   if str rel > 0 then goto string track out;
     arn(13e4) D               ;   inf 1:= str rel track;
     ga  2e4   , gm  3e4       ;   inf 2:= M;
     pm  8e4   , gm  6e4       ;   last track:= string track;
     pm  d14   DX              ;   output(vendpass);
     pm  5e4   , hs  e3        ;   input track:= first track
     gm  8e4   , hs  c53       ;   start medium:= pack medium;
     gr  7e4   , hhn e29       ;   go to end pass;
```

[4.3. cont.]

```
c23: hs  c7                    ; colon: comp or simple;
     ca d9 [t=], hvn a15       ;   if Raddr = t = then goto colon equal;
b18: pmd27[v:] DXV             ;   swap; Raddr:= colon value;
     [in code heading d16]     ;     goto double out;
a12: pm d28[v=]DX              ; equal1: swap; Raddr:= v = ;
a13: hs  e3              IZC   ; double out: normal:= t; commok:= f;
     hv  c4    X           ;   output (Raddr); swap; goto in normal;


c24: hs  c7                    ; equal: comp or simple;
     nc d43[t>], hv  a12       ;   if Raddr ≠ t> then goto equal 1;
     pm d31[v=>] DXV           ;   Raddr:= v => goto combined out;
c25:                           ;
b19: pm d16[v,]  DVX           ; comma: Raddr:= comma value
     [in code heading d27]     ;   goto combined out;
a14: pmd25[vfat,]DXV  LZ       ; fat com: Raddr:= v fat, ; goto combined out;
a15: pm d30[v:=] DXV  LZ       ; colon eq: Raddr:= v:= ; goto combined out;
a16: pm d32[v-,] DX   LZ       ; negate: Raddr:= v -, ;
     hv  e3           IZC      ; combined out: normal:= t; commok:= f;
                               ;   goto output;

c26: hs  c7                    ; minus: comp or single;
     ca d10[t,], hvn a16       ;   if Raddr = t ,  then goto negate;
     pm d29[v-]DXV             ;   swap; Raddr:= v - ;  goto double out;

a17: pm d23[v)]DX              ; right parent 1:
     hv  a13                   ;   swap; Raddr:= v )  ;  goto double out;

c27: hsn c2              IZA   ; right parent: normal:= f; next;
a18: ck  -2              NQB   ;   if -, letter or blind ∨ comp
     hh  a19             LO    ;     goto right parent 1;
     ck  2               NQB   ;
a19: hv  a17   , ck  -2        ;   if -,letter then goto right parent;
     hv  c27             LO    ;
a21: hs  c2                    ; in fat: next;
     nc  d6[t:], ud  a18       ;   if letter or blind ∧ -,comp then
     hv  a21             LO    ;     goto in fat;
     nc  d6[t:], hv  a23       ;   if Raddr ≠ t:  ∨ comp then goto fat error;
a22: hs  c2                    ; end fat: next;
     ca d35[t(]                ;   if Raddr = t ( ∧ -,comp then
     hvn a14             NQB   ;     goto fat com;
     ck  -4              NQB   ;   if blind ∧ -, comp then
     hv  a22             LO    ;     goto end fat;


a23: hs  c49                   ; fat error:
     pm (b10)   , qq  sd58     ;   mess1(≮<)<improper>≯, 0, 1); R:= table[char];
     hv  a11   X               ;   goto test sem;
```

[4.3. cont.]

```
c28: qqn(b17)  t   1   IZB ; begin: end level:= endlevel + 1;
     arn d24   D       IZA ;   Raddr:= v begin ; commok:= normal:= t;
     ps  c2 -1 , hv  e3    ;   set return (next); goto output;


c29: arn d15   D           ; lined digit: Raddr:= lined digit diff +
     ar (b10)  , hv  c6    ;     table char; goto normal out;
```

[5.1. cont.]

```
b a10, b2;                          ; begin block for code

c30: pm  d3    DX                   ; code: Raddr:= v code;
     pa  b16   t   d1                ;   sem value:= v begcode;
     pp  d27   , it  d16             ;   p:= v: ;  colon value:= v,  ;
                                     ;   goto code beg end;
a1:  pa  b18   t   d27 IQC           ; code end: colon value:= v: ; comp:= lined:= f;
     gp  b19   , hv  c6              ; code beg end: comma value:= p;
                                     ;   goto normal out;


c31: pm  e4    , gm  b2              ; code 1: code CR count:= CR count;
a2:  pi1.1+1.8+0.9 V 764             ; end code text: first on line:= t; inbracket:= f;
                                     ;   code comm:= t; goto set blind CR;
c36: pi 1.1+0.8t  765  NRB           ; code CR: if -, in bracket then
                                     ;     begin first on line:= t; code comm:= f end;
     pa  b13   V   d2                ; set blind CR: out CR:= cv blind CR;
                                     ;   goto code byte out;
c37: pi  1.8   t   1021              ; code sem: code comm:= t;

c32: hs  e3              NZ          ; code byte out: if R ≠ 0 then output;
c33: hsn c1              IZA         ; next code: normal:= f; next1;
     hv  a5              LQB         ;   if comp then goto code comp;
a3:  ck  20    , ga  b1              ; after code comp: code byte:= part 3(R);
b1:  hvn -1    t   c   IO            ;   if code byte < 0 then
     [code byte]                     ;     begin R:= 0; goto code action[codebyte] end;
     hv  c33             LRA         ;   if code comm then goto next code;
     arn(b1)   DV        IZB         ;   Raddr:= code byte; first on line:= f
                                     ;   goto finis code byte;
c34: pi  0     t  1020 LRB           ; code right bracket:
                                     ;   if in bracket then code comm:= in bracket:= f;
a10: pa  b13   t  d39  NZB           ; finis code byte: if -, first on line then
     hv  c32                         ;   out CR:= v CRcode; goto code byte out
c38: arn d4    D                     ; code colon: Raddr:= cv;
                                     ; code left bracket:
c35: hv  c33             LRA         ;   if code comm then goto next code;
     pi 1.8+1.9t  1020 LZ            ;   if R = 0 then code comm:= in bracket:= t;
     hv  c36                         ;   goto code CR;
```

[5.1. cont.]

```
                                  ; code comp:
a5:  hv  a3              LRA      ;   if code comm then goto after code comp;
     hv  a6              NQA      ;   if -, lined then goto code comp err;
     hv  a3              NZB      ;   if -, first on line then
                                  ;     goto after code comp;
     ca  4 [td],  hv c33          ;   if Raddr = td then goto next code;
     ca  5 [te], hv  a7           ;   if Raddr = te then goto finis code;
     ca 20 [tt], hv  a8           ;   if Raddr = tt then goto codetext;
     ca  6 [tf], psn d47          ;   if Raddr = tf then Raddr:= c f
     ca 13 [tm], psn d5           ;   else if Raddr = tm then Raddr:= c m
     pm  s     DXV     LZ         ;   else
a6:  pm  d36   DX                 ; code comp err: Raddr:= c forbidden;
     hh  a3                       ;   R := R shift -20; goto after code comp;

a7:  pm  d48   DX                 ; finis code:
     hs  e3                       ;   output(c e);
     pan b16   X   d12            ;   sem value:= v; ; M:= 0
     arn e4    , sr  b2           ;   MR pos 48:= CR count - code CR count;
     tl  -9    , pp  d16          ;   p:= v, ;
     hs  e5              NZ       ;   if R ≠ 0 then mess (‹<code›, 2, 0);
     ps  c2-1  , qq  nd59         ;   set return (next);
     pa  b13   X   d22            ;   out CR:= v CR;
     mt  4a30  , hv  a1           ;   Raddr:= - MR pos 48; goto code end;
b2:  qq [code CR count]           ;

a8:  qq  c55   , hs  c54          ; code text: init str (test code char);
     bs  p-41  , it  d38          ; after code str word: const kind:=
     pa  b24   t   d37            ;     if last word then cv string else cv last str;
     tl  -6    , hs  c56          ;   output 5 of code;
     bs  p-41  , hvn a2           ;   if last word then goto end code text;
     ps  a8    , hv  c48          ;   set return (after code str word);
                                  ;   goto get str word (test code char);

e                                 ; end block for code
```

[5.2. MEDIUM CHANGE AND INPUT DRIVERS]

```
b a27, b9, d1                ; begin

c40: arn 8e4   , hs  e11    ; Copy:
     sk  42e13 , vk  960     ;   to drum (string track, inbuf 2);
b27: can[search], hh  a2     ;   if search = 0 then goto copy err 1;
     vk (b27)  , lk  42e13   ;   from drum (search, inbuf 2);
     pa  b1    t   d49       ;   i name:= i name 0;
     gi  b2    , vk  960     ;   save in:= in;  wait drum;
     qq  c57   , hs  c54     ;   R:= init str (test copy char);
b1:  gr [iname], arn b1      ; rep name: store[i name]:= R;
     bs  p-41  , hv  a1      ;   if lastword then goto search name;
     ca  6d49  , hv  a3      ;   if i name = iname 0 + 6 then goto copy err 2;
     qq  c57   , hsn c48     ;   R:= get str word (test copy char);
     hv (b1)   D   1         ;   i name:= i name + 1; goto rep name;

a1:  pa  a10   , hs  c53     ; search name:  pack medium;
     pa  56e13 t  1e13       ;   place 0 in look up:= inbuf 1;
     qq (46e13)t   2         ;   free is treated as normal area;
     pp  d49-1 , hs  46e13   ;   p:= i name 0 - 1; search;
a2h: nc  0     , hs  e5      ;   if Raddr ≠ 0 then
b2:  pi [save in], qqn d66   ; copy err 1: mess (‹<copy›, 2, 0)
     arn 8e4   , hs  e11     ;   in:= save in; R:= areaword
     arn 44e13 , lk  42e13   ;   from drum (string track , inbuf 2);
     vk (e17)  , ps  c2-1    ;   wait drum; set return (next);
     mb  4a30          NT    ;   if R > 0 then clear bits(8,23,R);

c52:                         ;
b26: gr d50 [i copy] t -1    ; new medium:  stack[i copy-1]:= R;
     gs  b7    , it  0       ;   i copy:= i copy -2;  save s := s;

c41: pmn(b26)  X t 1         ; finis:  i copy:= i copy + 1;  R:= stack[i copy];
     tl  -7    , ga  b3      ;   kind:= bits(0, 2, R);
     tl  -25   , it (b3)     ;   if kind < -1 then goto copy err 1;
     bs  -1    , hh  a2      ;   char no:=
     tln 16    , tk  16      ;     bits(8,23,R) pos 23;
     gr  b20   , tln  16     ;   M:= bits (24,39,R);
b3:  pm [kind] XV t a25 NB   ; start medium:  if LB then
a3:  ps  a4    , hv  e5      ; copy err 2: mess(‹<stack›, 2, 0);
     ga  1c2   , gt  b11     ;   R:= descriptor[kind];
     gm  -2    , ck  20      ;   next word action:= part 1(R);
     ga  b4    , gt  b5      ;   action 63:= part 2(R);
a4=i-1                       ;   medium track:= M;  start action := part 4(R);
b4:  pm  -1    , qqn e34     ;   input instr 1 := instr [part 3(R)];
     gm  c2    , it  1       ;   input instr 2 := instr [part 3(R) + 1];
     pm (b4)   , gm  2c2     ;   M:= stack[i copy];  R:= 0;
b5h: pm (b26)  , hvn[start];   goto action [start];
```

[5.2. cont.]

```
c53: arn(b26)    , tl  -37    ; pack medium:  R:= stack[i copy];
     hhn a8                LT ;   if kind part > 3 then return;
     arn(e2)     DVt-1e13LZ   ;   if kind part = 0 then
     arn 5a30    , hv  a6     ; pack drum medium:
     pm  -2      , gm (b26)   ;     begin stack[i copy]:= medium track;
     ck  10      , hv  a7     ;     R:=inaddr - inbuf1; comment wordno;
a6:  il  0       , arn 1e13   ;     end
     mb  4a30    , gr (b26)   ;   else pack buf medium:
     arn 4e13    , ar  -5     ;     begin stack[i copy]:= area word from buf 1 ∧
a7:  pm (e1)     DXt d1       ;       8 m 16 0 16 m;
     ck  -14     , ml  13a30  ;     R:= block length from buf 4 - left in buf end;
a8h: xr          , ar (b26)   ;   stack[i copy]:= R:= stack[i copy] +
     gr (b26)    , hr  s1     ;     (R x 6 + byte address - charbuf0 + 1) pos 23;
                              ;   return;
                              ; start medium:
a10: hh  r8      , vk  960    ;   if called by copy then begin
b28: can [init med.], hh a2   ;   if init medium = 0 then goto copy err 1;
     vk (b28)    , lk  1e13   ;   init medium 1 to inbuf 1;
     arn 8e4     , hs  e11    ;   to drum(string track, inbuf 1);
[s-1]sk  42e13   , arn(b26)   ;   R:= stack[i copy];
     hs  1e13                 ;   init medium; comment init medium 2
                              ;     to inbuf 2;
     hh  a2                NZ ;   if R ≠ 0 then goto copy err 1;
     arn 8e4     , hs  e11    ;   from drum(string track, inbuf 2);
[8]  lk  42e13   , pmn r      ;   end;  M:= not zero;  R:= 0;

a11: ar  -2      , gr  -2     ; get medium track:
     hs  e11                  ;   medium track:= medium track + R;
     pa  e2      X t  e13     ;   from drum (medium track, inbuf 1);
     lk  1e13    , vk  (e17)  ;   in address:= inbuf 1 - 1; wait drum;
     hh  a13               NZ ;   if M ≠ 0 then goto skip to char;

     arn -2      , sr  5e4    ; check against input ouput overlap:
     hv  a15               LT ;   if (medium track ≥ first track ∧
     arn 9e4     , sr  -2     ;       medium track ≤ output track) ∨
     hv  r4                NT ;      (medium track ≥ stringtrack ∧
     arn -2      , sr  8e4    ;       medium track ≤ last track)
     hv  a15               LT ;   then
     arn 6e4     , sr  -2     ;     mess (‹<copy overlap›, 2, 0);
[4]  hs  e5                NT ;
     hv  a15     , qqn d62    ;   goto get drum word;

a12:                          ; skip to char:
a13h:gr  b20     , hs (1c2)   ;   get next word;
     arn b20     , sr  13a30  ;   if char no - 6 pos 23 ≥ 0 then
     hv  a12               NT ;     begin charno:= charno - 6; goto skip to char end;
     arn b20     , ck  14     ;   byte address:= byte address + char no;
     ac  e1      , arn(e1)    ;   s:= save s;  R:=store[byte address];
b7:  ps [save s], hv 1c2      ;   goto central input test if next word;
```

[5.2. cont.]

```
                                ; Next word from drum: M:= 0;
                                ;   if in address > inbuf 1 + 39 then
a14: bsn (e2)  X t  39e13       ;     begin  R:= 1; goto get medium track  end;
     arn 7a30  , hv a11         ; get drum word:  in address:= in address + 1;
a15: pmn (e2)  V t  1           ;   R:= 0;  M:= store[in address];  goto common;
a16: hs  21e13                  ; Next word from buf medium:  next buf word;
     pa  e1    V t   b8 LZ      ;   if R ≠ 0 then
     ps  r1    , hv  e5         ; copy err 3: mess(≪copy medium≫, 2, 0);
     arn  a19  , gr  a17        ;   set unpack instruction;
[2]  hvn  a18  , qqn  d67       ;   goto unpack;

b8:  qq  ; char buf 0:  coment a word containing
     qq  ;   6 characters is unpacked here by the
     qq  ;   instructions below.
     qq  ;   The loop stops when the sixth character
     qq  ;   overwrites the instruction in a17;
a17: zq [see 3a16]              ;
     arn (e1)  , hr  s1         ; unpack: unpack R and replace 63 by 64;
a18: cln -6                     ;   R:= store [byte address];  return;
     ca  63.5  , arn 7a30       ;
     ck  -4    , hv  a17        ;

a19: gr  b8-1  V t 1            ; unpack instruction.
d1 = -b8+1 [see 9c53]           ;
```

[5.2. cont.]

```
b a8, b3                    ; begin block for input from keys:

a20: hs  c49                ; start keys:
     hvn r1    , qq (pd63)  ;   mess 1(<type in>, 2, 3); R:= 0;
     bs (b14)  , it  60     ;   set part 1 (save case and 3 blinds,
     pa  b3    t   58       ;     if case 1 > 0 then 60 else 58);

a21: gp  b1    , ud  14e4   ; new key line: save p:= p; select(typewriter);

a1:  pp  e13   , gp  e1     ; after no: inaddress:= byte address := basein;
     gp  e2    , pp  -119   ;   p:= -119; if R ≠ 0 then from new keyline;
     ppn -159  V       NZ   ;   begin p:= -159; R:= 0 end
     arn a7    , sy 64 [CR] ;   else begin writecr; R:= four case end;

a2:  ca  64    , hh  a6     ; pack key: if Raddr = 64 then goto end line;
     ck  10    , bs  p      ;   R:= R shift 10; if p > 0 then
     gr  pe13  , pp  p-159  ;     begin inbuf 1[p]:= R; p:= p - 159 end;
     ly  b2    , pm  a8     ;   no clear Raddr:= type char; M:= four 63;
     pp  p40   , bs  p474   ;   p:= p + 40; if p < 38 ∧ R ≠ four case then
     cm  a7    , hv  a2     ;     goto pack key;

     sy 29[RED], sy 17[<]   ; stop line: write char (w RED); writechar(w <);
a3:  lyn b2    , bs  p474   ; read stop inf: Raddr:= typechar;
     ca 54[f]  , hh  a4     ;   if p < 38 ∧ Raddr = wf then goto finis;
     nc 37[n]  , hv  a3     ;   if Raddr = wn then goto read stop inf;
     sy 38[o]  , sy 62[blk] ; no: write char(wo); writechar (w BLACK);
                            ;   R:= 0; goto after no;
a4:  hvn a1    , sy 57[i]   ; finis: write char (wi); writechar(wn);
     sy 37[n]  , sy 57[i]   ;   writechar(wi); writechar(ws);
     sy 18[s]  , sy 62[blk] ;   write char (w BLACK);
     pt  b11   t   c41      ;   action 63:= finis;
     arn b3    , hh  a6     ;   R:= save case and 3 blinds; goto end line;

a5:  pp  p40   , tk  10     ; fill word: p:= p + 40; no clear R addr:= 63;
a6:  cl  -10   , ck  10     ; end line: R:= R shift 10;
     bs  p512  , hv  a5     ;   if p < 0 then goto fill word;

     gr  pe13  , gm p1e13   ;   inbuf 1[p]:= R; inbuf 1[p+1]:= M;
b1:  pp  -1    , ud  16e4   ;     comment give a 63-byte if no fill word;
     [save p]              ;   p:= save p; select (normal);
b2:  qq[for ly], hv  1c2    ;   goto next plus 1; comment LA;
                            ;     constants:
b3:  qq[save case]+127.19+127.29+127.39,; save case and 3 blinds
a7:  qq 58.9+60.19+58.29+60.39        ,; four case
a8:  qq 63.9+63.19+63.29+63.39        ,; four 63

e                           ; end block for input from keys
```

[5.2. cont.]

```
a23: tl  49     , ca  1      ; start ly medium: if bits(10,19,M) = 1 then
     hv (b3)    D t   a26    ;   begin kind:= -2; goto start medium end;
     arn 16e4   , ck  -23    ; start reader: normal input unit:=
     tk  23     , ar (b26)   ;   bits (17, 19, copy stack[i copy];
[-2] gt  16e4   , ud  16e4   ;
a24h:hv  c2     , ud  14e4   ; after pause: select(normal unit);
     lyn b10    , hh  r-2    ;   goto next;

c42: hs  c49                 ; end code: mess 1(≠<pause≠, 2, 3);
     hh  a24    , qqn pd65   ;   select (type writer); lyn; goto after pause;
```

[Input medium descriptors. Accessed through kind and unpacked in next medium.

```
Kind   Next word   63      Variable  Start
        action    action   instruct. action ]

[-2] qq  e2.9 +  a21.19 + 8a30.29 + a20.39 ; keys, accessed via kind -1
[-1] qq           c2.19 +10a30.29 + a23.39 ; reader
a25: qq a14.9 +   c2.19 + 8a30.29 + a11.39 ; drum
[1]  qq a16.9 +   c2.19 + 8a30.29 + a10.39 ; buffer medium
[2]  qq a16.9 +   c2.19 + 8a30.29 + a10.39 ; buffer medium
[3]  qq a16.9 +   c2.19 + 8a30.29 + a10.39 ; buffer medium
```

```
d a26= -1a25 [see 1a23];
e                        ; end block for medium change and input drivers;
```

[5.4. LAY OUT, comments on next page]

```
b a19, b9;                          ; begin block for layout:
                                    ; layout or string:
c44: qq  i+1   , hs  c54            ;   init str (here);
     ps  0     , ca d41[<]          ; here: s:= 0; if -, comp ∧ Raddr = t< then
     hv  c45           NQB          ;     goto string;
     pa  b24   X   d26              ;   const kind:= v layout; M:= 0;
     pan b1    X   -19              ;   position:= -19; space word:= 0;
     gm  b20   , hv  a13            ;   goto start layout;

a1:  ar  3a30                       ; lined plus: R:= R + sign bit;
a2:  ar  3a30                       ; plus:       R:= R + sign bit;
a3:  ar  3a30  , pp  p1             ; minus:      R:= R + sign bit; p:= p + 1;
     bs  s-6   , ck  -10            ;   if s > 6 then R:= R shifts - 10
     ac  1b20  , hv  a12            ;     comment exponent sign;
                                    ;   layout word:= layout word + R;
                                    ;   goto next layout;

a4:  pp  p7    , ps  6              ; point: p:= p + 7; s:= 6;
     qq (b1)   V   1                ;   position:= position + 1; goto clear check;
                                    ; ten: p:= 13; s:= 12;
a5:  pp  13    , ps  12             ; clear check: clear part 1(layout word);
     pa  1b20  , hv  a12            ;   goto next layout;

a6:  bs (b1)   , hv  a11            ; space: if position  > 0 then goto set alarm;
     arn 7a30  , ns (b1)            ;   space word:= space word + bit(19-position);
     ck  s-20  , ac  b20            ;   R:= 0;
     can p     , hv  a10            ;   if p = 0 then goto increment
     pp  p+1   , hv  a12            ;   p:= p + 1; goto next layout;

a7:  arn 2a30  V                    ; n: R:= n bit; goto d;
a8:  srn 1a30  , is  s2             ; zero: R:= - b bit; p:= s + 5 goto commond;
a9:  pp  s3    , ar  sa30           ; d: p:= s + 3;
                                    ; common d: R:= R + d increment [s];
a10: ac  1b20           IOA         ; increment: layout word:= layout word + R;
b1:  qq  -1    V   1    NOA         ;   if bit (0, layout word) = 0 then
     [position]                     ;     position:= position + 1;
a11: pp  16                         ; set alarm: p:= 16; comment error state;
a12: hsn c1             IZA         ; next layout : normal:= f; next 1;
a13: hh  a14            NQB         ; start layout: layout class:=
     pa  b2    t   10   NQA         ;     if comp then
     nc  d43[>]V        NQA         ;        (if -, lined ∧ Raddr = t > then 10
     ca  d42[+], it  9              ;         else if lined ∧ Raddr = t+ then 9
     pa  b2    , ca  d10[,];        ;         else if lined ∧ Raddr = t, then 5
     pa  b2    t   5    LQA         ;         else 0)
a14: hv  a15   , ck  -10            ;     else bits(30, 33, R);
     tk  -6    , ga  b2             ;
a15:                                ;   R:= layout table [layout class];
b2:  arn -1    t a17    IZA         ;   layout act:= Raddr; normal:= t;
     [layout class]                 ;   if bit (p + 10, R) = 1 then
     ga  b3    , ck  p10            ;     begin R:= 0; goto layout act end;
b3:  hvn[layout act]    LO          ;   if -,mark A then
     hv  a11            NA          ; not terminator: goto set alarm;
     pp  16    , hv  a16            ;   p:= 16; goto terminator;
```

[5.4. cont.]

```
a16: pa  1b20  , arn 1b20  ; terminator: clear part1(layout word);
     ac  b20   , tk  19    ;   space word:= space word + layout word;
     pp  16          LØ    ;   if b overflow then p:= 16;
     bs  p-15  , hs  c49   ;   if p > 15 then mess1(‹<string›, 0, 1);
     hv  c46   , qq  sd69  ;   goto const out;
```

[During layout scanning p holds the current state (0-16) and s holds a function of this state indicating before point (0), after point (6), or, after $_{10}$ (12). s is used to set p and to reference the proper constant which is added to the layout word for d, r, or, 0. Initially, s = p = 0
The actions are governed by a state table which contain one word for each allowed layout symbol and a common word (word 0) for forbidden ones.
Each word consists of an action address in bit 0 to 9 and check bits in bit 10 to 26, corresponding to state 0 to 16, A one indicate that the symbol is allowed in the corresponding state.
The table is referenced via bits 30 to 33 of the main table which gives the address relative to first word (slipname a17) or, for compound symbols, from the code.

```
      action.  state bits.  symbol  allowed in states.      new states.
                                                             p    s]
a17: qq                    ; forbidden                       16   -
[1]  qq a2   t 1.0+1.13    ;   +       0,13                  p+1  -
[2]  qq a3   t 1.0+1.13    ;   -       0,13                  p+1  -
[3]  qq a4   t 53.5        ;   .       0,1,3,5               p+7  6
[4]  qq a5   t 5.5+5.11    ;   10      3,5,9,11              13   12
[5]  qq a6   t 37.5+5.11   ;   ±       0,3,5,9,11            see code
[6]  qq a7   t 3.1         ;   n       0,1                   s+3  -
[7]  qq a8   t 15.6+15.12  ;   0       3,4,5,6,9,10,11,12    s+5  -
[8]  qq a9   t 217.7+231.15;   d    0,1,3,4,7,8,9,10,13,14,15 s+3  -
[9]  qq a1   t 1.0+1.13    ;   +       0,13                  p+1  -
[10] qq a16,qq 5.5+163.16  ;   ⏋       3,5,9,11,15,16   terminates layout
                           ;     Note ,mark.
```

[Formation of layout: Bits 20 - 39 of the final layout is accumulated in word 1, the layout word. Format:
dh-check.4 or s-check.3 + b overflow.19 + b.23 + h.27 + fn.29 + d.33 + n.34 + s.37 + fe.39.
dh-check or s-check is incremented and tested for each d, zero, or n in the layout input an gives alarm when d > 15 ∨ h > 15 ∨ s > 7
b overflow is tested when the layout is completed and gives alarm if b > 15.
Bits 0 - 19 of the layout is accumulated in word 0, the spaceword, and the final layout is formed here before it is output]

```
e                          ; end layout
```

[5.5. cont.]

```
b a10, b1                    ; begin block for string

c45: pa  b24   t   d20       ; string: const kind:= vshort string;
b21: pa[string level]Dt 1    ;   stringlevel:= 1;
     qq  a5    , hs  c48     ;   get str word (test str char);
     bs  p-41  , hh  c46     ;   if last wrd then goto end const
                             ;
     pm (13e4) DX            ; set long string description:
     ck  10    , gr  b20     ;   word 0:= str rel track + (str rel shift 20);
     it (14e4) , pt  b20     ;   const kind:= vlong string;
     pa  b24   XV  d21       ;   go to store string:

a1:  qq  a5    , hs  c48     ; next string word: get str word(test str char);

     is (14e4) , gr s42e13   ; store string: str buf [str rel]:= M;
     arn(14e4) D    1        ;   str rel:= str rel + 1;
     nc  40    , hv  a2      ;   if str rel = 40 then
                             ;     begin string track out:
c51: arn 8e4   , hs  e11     ;     to drum (string track, str buf);
     sk  42e13 , pa  14e4    ;     str rel:= 0;
     srn 7a30  , ac  8e4     ;     string track:= string track - 1;
                             ;     available tracks:= available tracks - 1;
     arn 4e4   , sr  7a30    ;     if used tracks > available tracks then
     gr  4e4   , sr  1e4     ;
     qq (13e4) t   -1        ;         mess(<<program too big>);
     hs  e5            LT    ;     str rel track:= str rel track - 1;
     vk (e17)  , qqn e33     ;     wait track
a2:                          ;     end;
b22: bs p470   , hv  a1      ;   if -, last word then goto next string word;
                [see finis pass 1]
c46: arn b20   , hs  a10     ; const out: R:= word 0;
     ps  c2-1  , hv  c1      ; end const: output 5; set return (next)
                             ;   goto next 1;

c56: pt  b1    t   -1        ; output 5 of code: cut := -1;
a9:                          ; next of 5:
b1:  tl  10    , ck 0 [cut]  ;   RM:= RM shift 10; R:= R shift cut;
a10:                         ; output 5:
b24: pm [const kind] DX      ;   swap; Raddr:= const kind;
     hs  e3                  ;   output (Raddr); swap;
     ncn 0[i]  X   256       ;   i:= (i+1) mod 4; comment i starts at 0;
     ga  b24   , hv  a9      ;   if i ≠ 0 then
     pt  b1    , hv  e3      ;     begin const kind:= Raddr; goto next of 5 end
                             ;   cut:= 0; goto output;
```

[5.5. cont.]

```
c54: grn 1b20  , pp  0      ; init str: word 1:= p:= 0;
     pa  b25   t   -1  IZA ;   str case:= -1; normal := f;

c48: grn 2b20               ; get str word: word 2:= 0

a3:  hs  c1               LZA ; next str char: if -, normal then
     hv  (s)              LZA ;     begin next 1; goto char test [s] end;
a4:  arn d19   DX         IZA ; terminator: Maddr:= s⟩; R:= bit 37; normal:= t;
     arn 12a30 , hv  a7     ;   goto pack it;

a5:  hv  a7               NQB ; test str char: if -, comp then goto pack it;
     nc d43[>] , hh  a6     ;   if Raddr = t> ∧ -, lined then
     bt (b21)  t   1   NQA ;     begin stringlevel:= stringlevel + 1;
     hv  a4            NQA ;     if stringlevel > 1 then goto terminator end;

c57: hh  a6               LQB ; test copy char: if comp then goto comp char;
     ca d11[SP], hv  a3     ;   if Raddr = tSP then goto next str char;
     ca d41[<] , hv  a4     ;   if Raddr = t< then goto terminator;
     hv  a7                 ;   goto pack it;

c55: hh  a6               LQB ; test code char: if comp then goto comp char;
     ca  d8[;] , hv  a4     ;   if Raddr = t; then goto terminator;
                            ;   goto pack it;
a6:  hv  a7    , ca d10[,] ; comp char: if Raddr = t, ∧ lined then
     hvn a7    X       LQA ;     begin R:= 0; swap; goto pack it end;
     gm  3b20  , ca d41[<] ;   word 3:= M; if Raddr = t< ∧ -, lined then
     hs  c43           NQA ;     string nest (⟨<⟨in string⟩ , 0, 3);
     pm  12a30 , qq  pd60  ;   Maddr:= s _;
     arn d17   DX          ;   R:= if lined then bit 37 else bit 39;
     arn  7a30         NQA ;
     hs  c47               ;   pack char;
     arn(b10)  , pm  3b20  ;   M:= word 3; R:= table[char];
                           ;
a7:  hs  c47               ; pack it: pack char;
     bs  p475  , hv  a3    ;   if p ≤ 36 then goto next str char;
     pm  2b20  , arn 1b20  ;   R:= word 1; word 1:= word 2;
     gm  1b20  , bs  p469  ;   if p ≤ 42 ∧ normal then last word;
     ar  d19.3 DV      NZA ;     R:= R + s⟩ shift 36
     ar  d40.3 DV          ;   else
     hr  s1                ;     begin R:= R + s wordend shift 36
     pp  p-36  , hr  s1    ;       p:= p - 36 end;
                           ;   return;

c47:
b25: ck  -1    X       IOB ; pack char: set case:= bit(40+str case, R) = 1;
     [str case]            ;   swap; if set case then
     pm (b25)  DX  61  LOB ;     begin swap; R:= 0; Raddr:=
                           ;       str case:= str case + 61 end;
a8:  ck  p-30  , bs  p481  ; pack after case: R:= R shift p - 30;
     ac  1b20  V           ;   if p ≤ 30 then word 1:= word 1 + R
     ck  4     , ac  2b20  ;   else word 2:= word 2 + (R shift 4);
                           ;   p:= p + 6;
     srn(b25)  DV -57  LOB ;   if -, set case then return;
     pp  p6    , hr  s1    ;   str case:= 57 - str case; swap;
     ga  b25   X       IZB ;   set case:= f; goto pack after case;
     pp  p6    , hv  a8    ;

e                            ; end block for string;
```

[6. MESS 1 AND LOADING OF TEXTS]

```
c43: qq (b21)  t  -1          ; string nest: stringlevel:= stringlevel - 1;
c49: arn 8e4              IPA ; mess 1: to drum (string track, str buf);
     hs  e11                  ;   print:= f;
     sk  42e13 , hv  e5       ;   goto mess;



b k=e31,  i=0                 ; load messages:
i=e32                         ;

d54: tcharacter;              ;
d55: toff;                    ;
d56: ton;                     ;
d57: tcomment;                ;
d58: t)<improper>.;           ;
d59: tcode;                   ;
d60: t( in string;            ;
d61: tcompound;               ;
d62: tcopy overlap;           ;
d63: ttype in;                ;
d65: tpause;                  ;
d66: tcopy;                   ;
d67: tcopy medium;            ;
d68: tsum;                    ;
d69: tstring;                 ;

e32=i                             ;
e                                 ;
```

[7. TABLES.

The pass 1 input table consists of two areas:
Input table lower case, starting at d50 and
Input table upper case, starting at d51.
Each area holds in principle one word for each possible character input value from 0 to 127.; however, as all proper input characters have values from 0 to 64, only these values are represented explicitly in the tables, the rest beeing represented by words belonging to the code of pass 1. This is possible because the admissible tablewords all are flagged with an f-mark and no word of the code is f-marked.

Format of tablewords:
3 different formats, characterized by the flagbits alone are used:

1. Not f-marked: Totally forbidden characters. The rest of the word may contain anything.

2. f-marked and comma-marked: The word describes a central action, i.e. an action which is performed by the central input mechanism independent of context; e.g. UC, LC, CR, _ |
The action itself is specified by a jump instruction in the left half of the tableword.
The word also contains the boolean no printing (see below).

3. f-marked and not comma-marked: The word contains 5 parts as follows:

Part1.9
Action (>511, relative to c) or output value (≤511) in normal mode, i.e. outside compound symbols, code, layouts, strings, and, comments. Part 1 is also used for recognization of specific characters in other modes.

Part 2.19
Compound table reference. When an underlined or barred character is met in normal mode, part 2 is used as index to a table which describes compound symbols with this first character.

Part 3.29
Action (>511, relative to c) or output value (≤511) in code.

Part 4.33
Reference to the action table for layouts.

Part 5.39
Part 5 contains six booleans (1=true):
Skipped in compound error.34:
Is skipped, even if not underlined, during the reading of the rest of a compound symbol.
No printing.35:
Some characters, e.g. TAB, are not printed.
Blind in a construction which may be a fat comma.36
Requires lower case in strings.37
Part of a fat comma, i.e. letters.38
Requires upper case in strings.]

[7. cont.]

```
d51=i           ; Input table upper case[0].
c50=651         ; reserve input table upper case[0:64].
i=d50           ; start block at input table lower case [0]

b  a15, b5                  ; begin compound and input table.

[procedure compound; comment is called by hs c50 with comp = t and
 main table word in R.
 Exit:  hr s1 with: comp = lined = normal = f.
        R contains compound table word, or, in case of unidentified
        compound, main table word of first non-compound character.
        Possible error message has been given.]

i = c50                     ; begin
c50: ck  10    , ga  b2     ; comp index:= part 2(R);
     hv  a1             LQA ; if -, lined then
                            ;   begin comp index:= comp index + 1
     arn(b2)   t   1   IQB ;   R:= table[comp index]; comp:= f;
     pa  b2    t  d96  NA  ;   if -,LA then comp index:= undef bar comp;
                            ;   end;
a1:  arn(b2)   , hv  a6     ; R:= table[comp index]; goto comp action;

a2:  hv  a9            NQA ; check lined: if -, lined then
                            ;   goto comp undefined;
a3:  hsn c1            IZA ; next comp in: normal:= f; next 1;
     ga  b3    , arn(b2)    ;   test char:= Raddr; R:= table[comp index];
a4:  ck  20    , nc (b3)    ; compare next: R:= R shift 20;
                            ;   if Raddr ≠ test char then
a5:                         ; next comp word:
b2:  arn -1    t   1        ;   begin comp index:= comp index + 1;
    [comp index]            ;   R:= table comp index;
                            ;   end;
a6:  gt  r     , hv  -1     ; comp action: goto part 2(R)
                            ;
a7:  hsn c1            IZA ; comp in must agree: normal:= f; next 1;
     ga  b3    , arn(b2)    ;   test char:= Raddr; R:= table[comp index];
                            ;
a8:  ck  20                 ; compare must agree: R:= R shift 20;
b3:  ca  -1    , hv  a6     ;   if Raddr = test char then goto comp action;
     [test char]            ;
a9:  hsn c1            IZA ; comp undefined: normal:= f; next 1;
     hv  a9            LQB ;   if comp then goto comp undefined;
a10: pa  b2    t   b20      ;   comp index:= addr of (word 0);
     gr  b20   , hv  a14    ;   word 0:= R; goto comp error;
```

[7. cont.]

```
a11: hsn a15          IZA ; comp rest 7: rest;
[1]  hsn a15          IZA ; comp rest 6: rest;
[2]  hsn a15          IZA ; comp rest 5: rest;
[3]  hsn a15          IZA ; comp rest 4: rest;
[4]  hsn a15          IZA ; comp rest 3: rest;
[5]  hsn a15          IZA ; comp rest 2: rest;
[6]  hsn a15          IZA ; comp rest 1: rest;
[7]  hv  a14          NQA ; comp rest 0: if -, lined then goto comp err;
a12: arn(b2)          IQC ; finis comp: R:= table[comp index];
a13:                      ;   lined:= comp:= f;
b4:  bs  1     , hr  s1   ; exit comp: if -, rest err then return;
     [rest err]           ;   comment > 0 = false;
a14: pa  b4    t   1      ;   rest err:= f;
     hs  c49              ; comp err: mess 1(<<compound>, 0, 1);
     hv  a12   , qq  sd61 ;   goto finis comp;
a15: hv  c1          LQA ; rest: normal:= f;
                          ;   if lined then goto next 1;
     arn(b10)  , ck  -6   ;   if bit(<letter>, table[char]) = 0 then
     hr  a14        NO    ;     return to comp err;
     pa  b4    , hv  c1   ;   comp alarm:= t; goto next 1;
                          ; end;
```

[Redifinition of actions in normal mode so that they are negative
and relative to c = c44 + 1 (last normal action)]

```
c = c44+1,  c9= c9-c, c18=c18-c, c19=c19-c, c20=c20-c, c21=c21-c
c22=c22-c, c23=c23-c, c24=c24-c, c25=c25-c, c26=c26-c, c27=c27-c
c28=c28-c, c29=c29-c, c30=c30-c, c32=c32-c, c33=c33-c, c34=c34-c
c35=c35-c, c36=c36-c, c37=c37-c, c38=c38-c, c40=c40-c
c41=c41-c, c44=c44-c
```

[Definition of d names which are used to test table values which are
actions in normal mode]

```
 d6=c23[:],  d7=c22[end],  d8=c21[;],  d9=c24[=]          ;
d10=c25[,], d11=c19[SP],  d45=c18[message],  d46=c2[next]-c;
```

[7.1. COMPOUND TABLE]

```
     [  OUTPUT     COMP.      COMP.      COMP.       COMP        OUTPUT
        VALUE      ACTION     CHAR       ACTION      SYMBOL      VALUE
        or         BEFORE     TO         WHEN                    WHEN
        NORMAL     CHAR       TEST       CHAR                    PART 1
        ACTION     TEST                  FOUND                   IS
        NAME                                                     ACTION]

d:   qq d46.9 + a9.19                                  ;  undefined lined, skip comp
[1]  qq d46.9 +a10.19                                  ;       -       -  , single char
d75: qq 279.9 +a12.19                                  ;  =
[1]  qq 276.9 +a13.19                                , ;  ≠
[2]  qq 272.9 +a12.19                                  ;  <
[3]  qq c44.9 +a13.19                                , ;  ≮          d20, d21, d26
[4]  qq 277.9 +a14.19                                  ;  >
[5]  qq 269.9 +a13.19                                , ;  ≯
[6]  qq 274.9 +a12.19                                  ;  ≥
[7]  qq 270.9 +a12.19                                  ;  :

d76: qq c29.9 +a12.19                                  ;  <digit>        d41

d77: qq 245.9 + a3.19 +   2[b].29 + 6a11.39            ;  abs
     qq 144.9 + a8.19 +  18[r].29 + 4a11.39            ;  array

d78: qq c28.9 + a3.19 +   5[e].29 + 4a11.39            ;  begin          d24
     qq 130.9 + a8.19 +  15[o].29 + 2a11.39            ;  boolean
d79: qq 130.9 + a7.19 +  15[o].29 + 2a11.39            ;  Boolean

d80: qq 256.9 + a3.19 +   1[a].29 + 5a11.39            ;  case
     qq         a8.19 +  15[o].29 +   a5.39            ;  co..
     qq c30.9 + a2.19 +   4[d].29 + 6a11.39            ;  code           d3
     qq c20.9 + a4.19 +  13[m].29 + 3a11.39            ;  comment        none
     qq c40.9 + a4.19 +  16[p].29 + 6a11.39            ;  copy           none
     qq 251.9 + a8.19 +  18[r].29 + 6a11.39            ;  core

d81: qq 243.9 + a3.19 +  15[o].29 + 7a11.39            ;  do
d82: qq d34.9 + a3.19 +  12[l].29 + 5a11.39            ;  else
     qq         a8.19 +  14[n].29 +   a5.39            ;  en..
     qq c22.9 + a2.19 +   4[d].29 + 7a11.39            ;  end            d13
     qq 262.9 + a8.19 +  20[t].29 + 4a11.39            ;  entier

d83: qq 266.9 + a3.19 +   1[a].29 + 4a11.39            ;  false
     qq d33.9 + a4.19 +  15[o].29 + 6a11.39            ;  for
     qq c41.9 + a8.19 +   9[i].29 + 4a11.39            ;  finis           none

d84: qq         a7.19 +  15[o].29 +   a5.39            ;  go..
     qq d44.9 + a2.19 +  20[t].29 + 6a11.39            ;  goto
     qq         a8.19 + c19[ ].29 +   a5.39            ;  go . . or go_ . .
     qq d44.9 + a7.19 +  20[t].29 + 6a11.39            ;  go to or go_to
```

[7.1. cont.]

```
     [  OUTPUT     COMP.      COMP.      COMP.        COMP          OUTPUT
        VALUE      ACTION     CHAR       ACTION       SYMBOL        VALUE
        or         BEFORE     TO         WHEN                       WHEN
        NORMAL     CHAR       TEST       CHAR                       PART 1
        ACTION     TEST                  FOUND                      IS
        NAME                                                        ACTION]

d85: qq 106.9 + a3.19 +   6[f].29 + 7a11.39    ; if
     qq 116.9 + a8.19 +  14[n].29 + 2a11.39    ; integer

d86: qq 155.9 + a7.19 +   1[a].29 + 4a11.39    ; label

d87: qq c18.9 + a3.19 +   5[e].29 + 2a11.39    ; message        none
     qq 281.9 + a8.19 +  15[o].29 + 6a11.39    ; mod

d88: qq 258.9 + a3.19 +   6[f].29 + 7a11.39    ; of
     qq 109.9 + a8.19 +  23[w].29 + 6a11.39    ; own

d89: qq 137.9 + a7.19 +  18[r].29 +  a11.39    ; procedure

d90: qq 123.9 + a3.19 +   5[e].29 + 5a11.39    ; real
     qq 260.9 + a8.19 +  15[o].29 + 4a11.39    ; round

d91: qq 282.9 + a3.19 +   8[h].29 + 4a11.39    ; shift
     qq 149.9 + a4.19 +  23[w].29 + 3a11.39    ; switch
     qq         a8.19 +  20[t].29 +   a5.39    ; st ..
     qq 196.9 + a2.19 +   5[e].29 + 6a11.39    ; step
     qq 153.9 + a8.19 +  18[r].29 + 4a11.39    ; string

d92: qq 231.9 + a3.19 +   8[h].29 + 5a11.39    ; then
     qq 265.9 + a8.19 +  18[r].29 + 5a11.39    ; true

d93: qq 198.9 + a7.19 +  14[n].29 + 4a11.39    ; until

d94: qq 157.9 + a7.19 +   1[a].29 + 4a11.39    ; value

d95: qq 200.9 + a7.19 +   8[h].29 + 4a11.39    ; while

d96: qq d46.9 +a14.19                          ; undef bar comp
                                               ;
d74:   ; DEFINE LAST INSTR PASS 1              ;
```

[7.2. INPUT TABLE LOWER CASE]
i=d50

```
[ Normal  Comp.    Code     Layout  Skip in comperr  U   Input      Output
  output  table    output   table    No printing     n   value      value
  or      refer.   or       index     Blind in fat,  d   un =       when
  action  d=err    action   0=err       LC in string e   unused     part 1
  If central                             Ok in fat,  f              is an
  action then ,                           UC in str. .              action ]

qq c19.9 +  1d.19 + c33.29 + 5.33 + [fftftf=] 10.39   f ; 0  BLANK   none
qq  58.9 + d76.19 +  65.29 +        [ffftff=]  4.39   f ; 1  1
qq  59.9 + d76.19 +  66.29 +        [ffftff=]  4.39   f ; 2  2
qq  60.9 + d76.19 +  67.29 +        [ffftff=]  4.39   f ; 3  3
qq  61.9 + d76.19 +  68.29 +        [ffftff=]  4.39   f ; 4  4
qq  62.9 + d76.19 +  69.29 +        [ffftff=]  4.39   f ; 5  5
qq  63.9 + d76.19 +  70.29 +        [ffftff=]  4.39   f ; 6  6
qq  64.9 + d76.19 +  71.29 +        [ffftff=]  4.39   f ; 7  7
qq  65.9 + d76.19 +  72.29 +        [ffftff=]  4.39   f ; 8  8
qq  66.9 + d76.19 +  73.29 +        [ffftff=]  4.39   f ; 9  9
qq                                                    t ; 10 un
hv  c2         , qq                 [-f----=]         f ; 11 STOP    none
hv c42         , qq                 [-f----=]         f ; 12 END     none
qq  c9.9 +   d.19 +  32.29 +        [fffttf=]  6.39   f ; 13 aa      none
hv c12         , qq                 [-f-t--=]  4.29   f ; 14        none
qq                                                    t ; 15 ¯un
qq  57.9 + d76.19 +  64.29 + 7.33 + [ffftff=]  4.39   f ; 16 0
qq d41.9 +2d75.19 +  32.29 +        [ffftff=]  4.39   f ; 17 <
qq  19.9 + d91.19 +  41.29 +        [tffttf=] 38.39   f ; 18 s
qq  20.9 + d92.19 +  42.29 +        [tffttf=] 38.39   f ; 19 t
qq  21.9 + d93.19 +  43.29 +        [tffttf=] 38.39   f ; 20 u
qq  22.9 + d94.19 +  44.29 +        [tffttf=] 38.39   f ; 21 v
qq  23.9 + d95.19 +  45.29 +        [tffttf=] 38.39   f ; 22 w
qq  24.9 +   d.19 +  46.29 +        [tffttf=] 38.39   f ; 23 x
qq  25.9 +   d.19 +  47.29 +        [tffttf=] 38.39   f ; 24 y
qq  26.9 +   d.19 +  48.29 +        [tffttf=] 38.39   f ; 25 z
qq                                                    t ; 26 un
qq c25.9 +   d.19 +  30.29 +        [ffftff=]  4.39   f ; 27 ,      d16
hv c17         , qq                 [-t----=] 16.29   f ; 28 CLEAR   none
qq                                                    t ; 29 RED
hv  c2         , qq                 [-t----=] 16.29   f ; 30 TAB     none
hv c14         , qq                 [-t----=] 16.29   f ; 31 P OFF   none
qq c26.9 +   d.19 +  28.29 + 2.33 + [ffftff=]  4.39   f ; 32 -      d29
```

[7.2. cont.]

```
 [ Normal  Comp.    Code     Layout  Skip in comperr  U  Input      Output
   output  table    output   table    No printing     n  value      value
   or      refer.   or       index     Blind in fat,  d  un =      when
   action  d=err    action   0=err       LC in string e  unused     part 1
   If central                             Ok in fat,  f             is an
   action then ,                         UC in str.  .             action ]

qq  10.9 +   d.19 +  58.29 +          [tffttf=] 38.39  f ; 33  j
qq  11.9 +   d.19 +  33.29 +          [tffttf=] 38.39  f ; 34  k
qq  12.9 + d86.19 +  34.29 +          [tffttf=] 38.39  f ; 35  l
qq  13.9 + d87.19 +  35.29 +          [tffttf=] 38.39  f ; 36  m
qq  14.9 +   d.19 +  36.29 +          [tffttf=] 38.39  f ; 37  n
qq  15.9 + d88.19 +  37.29 +          [tffttf=] 38.39  f ; 38  o
qq  16.9 + d89.19 +  38.29 + 6.33 +   [tffttf=] 38.39  f ; 39  p
qq  17.9 +   d.19 +  39.29 +          [tffttf=] 38.39  f ; 40  q
qq  18.9 + d90.19 +  40.29 +          [tffttf=] 38.39  f ; 41  r
qq                                                     t ; 42  un
qq  28.9 +   d.19 +  32.29 +          [tffttf=] 38.39  f ; 43  ø
hv c15         , qq                   [-t----=] 16.29  f ; 44  P CN    none
qq                                                     t ; 45  un
qq                                                     t ; 46  un
qq                                                     t ; 47  un
qq  27.9 +   d.19 +  32.29 +          [tffttf=] 38.39  f ; 48  æ
qq   1.9 + d77.19 +  49.29 +          [tffttf=] 38.39  f ; 49  a
qq   2.9 + d78.19 +  50.29 +          [tffttf=] 38.39  f ; 50  b
qq   3.9 + d80.19 +  51.29 +          [tffttf=] 38.39  f ; 51  c
qq   4.9 + d81.19 +  52.29 + 8.33 +   [tffttf=] 38.39  f ; 52  d
qq   5.9 + d82.19 +  53.29 +          [tffttf=] 38.39  f ; 53  e
qq   6.9 + d83.19 +  54.29 +          [tffttf=] 38.39  f ; 54  f
qq   7.9 + d84.19 +  55.29 +          [tffttf=] 38.39  f ; 55  g
qq   8.9 +   d.19 +  56.29 +          [tffttf=] 38.39  f ; 56  h
qq   9.9 + d85.19 +  57.29 +          [tffttf=] 38.39  f ; 57  i
hv  c2         , qq                   [-f----=]        f ; 58  LC      none
qq 67.9 +    d.19 +  27.29 + 3.33 +   [fffttf=]  4.39  f ; 59  .
hv c11         , qq  d51              [-f----=]        f ; 60  UC      none
hv c16         , qq                   [-t----=] 16.29  f ; 61  SUM     none
qq                                                     t ; 62  BLACK
hh b11         , qq                   [-t----=] 16.29  f ; 63  TF      none
hv c10         , qq                   [-f----=]        f ; 64  CR      (b13)
d52: [secondary table words, used by the actions]        ;
qq d46.9 +   d.19 +  32.29 +          [---t-f=]  4.39     ; 65          none
qq d46.9 +   d.19 +  32.29 +          [---f-t=]  1.39     ; 66  ⊤       none
qq d46.9 +  1d.19 + c36.29 +          [f-tftf=] 10.39     ; 67  CR      none
```

[7.3. INPUT TABLE UPPER CASE]
i=d51

[Normal output or action If central action then ,	Comp. table refer. d=err	Code output or action	Layout table index 0=err	Skip in comperr No printing Blind in fat, LC in string Ok in fat, UC in str.		Undef	Input value un = unused		Output value when part 1 is an action]
qq c19.9 +	1d.19 +	c33.29 +	5.33 +	[fftftf=]	10.39	f ;	0	BLANK	none
qq 278.9 +	d.19 +	32.29 +		[ffffft=]	1.39	f ;	1	∨	
qq 267.9 +	d.19 +	32.29 +		[ffffft=]	1.39	f ;	2	×	
qq 268.9 +	d.19 +	32.29 +		[ffffft=]	1.39	f ;	3	/	
qq c24.9 +	d75.19 +	75.29 +		[ffffft=]	1.39	f ;	4	=	d28
qq c21.9 +	d.19 +	c37.29 +		[ffffft=]	1.39	f ;	5	;	d12
qq 210.9 +	d.19 +	c35.29 +		[ffffft=]	1.39	f ;	6	[	
qq 202.9 +	d.19 +	c34.29 +		[ffffft=]	1.39	f ;	7	]	
qq d35.9 +	d.19 +	31.29 +		[ffffft=]	1.39	f ;	8	(	
qq c27.9 +	d.19 +	78.29 +		[ffffft=]	1.39	f ;	9	)	d23
qq						t ;	10	un	
hv c2	, qq			[-f----=]		f ;	11	STOP	none
hv c42	, qq			[-f----=]		f ;	12	END	none
qq c9.9 +	d.19 +	32.29 +		[ffffft=]	3.39	f ;	13	AA	none
hv c13	, qq			[-f---t=]	1.29	f ;	14	|	none
qq						t ;	15	un	
qq 277.9 +4	d75.19 +	32.29 +		[ffffft=]	1.39	f ;	16	∧	
qq d43.9 +6	d75.19 +	32.29 +		[ffffft=]	1.39	f ;	17	>	
qq 47.9 +	d.19 +	9.29 +		[ffffft=]	3.39	f ;	18	S	
qq 48.9 +	d.19 +	10.29 +		[ffffft=]	3.39	f ;	19	T	
qq 49.9 +	d.19 +	11.29 +		[ffffft=]	3.39	f ;	20	U	
qq 50.9 +	d.19 +	12.29 +		[ffffft=]	3.39	f ;	21	V	
qq 51.9 +	d.19 +	13.29 +		[ffffft=]	3.39	f ;	22	W	
qq 52.9 +	d.19 +	14.29 +		[ffffft=]	3.39	f ;	23	X	
qq 53.9 +	d.19 +	15.29 +		[ffffft=]	3.39	f ;	24	Y	
qq 54.9 +	d.19 +	16.29 +		[ffffft=]	3.39	f ;	25	Z	
qq						t ;	26	un	
qq 68.9 +	d.19 +	32.29 +	4.33 +	[ffffft=]	1.39	f ;	27	⌧	
hv c17	, qq			[-t----=]	16.29	f ;	28	CLEAR	none
qq						t ;	29	RED	
hv c2	, qq			[-t----=]	16.29	f ;	30	TAB	none
hv c14	, qq			[-t----=]	16.29	f ;	31	P OFF	none
qq d42.9 +	d.19 +	29.29 +	1.33 +	[ffffft=]	1.39	f ;	32	+	

[7.3. cont.]

```
 [ Normal  Comp.    Code     Layout  Skip in comperr  U   Input      Output
   output  table    output   table    No printing     n   value      value
   or      refer.   or       index     Blind in fat,  d   un =       when
   action  d=err    action   0=err      LC in string  e   unused     part 1
   If central                           Ok in fat,    f              is an
   action then ,                         UC in str.   .              action ]

qq  38.9 +   d.19 +  26.29 +        [fffftt=]  3.39   f ; 33  J
qq  39.9 +   d.19 +   1.29 +        [fffftt=]  3.39   f ; 34  K
qq  40.9 +   d.19 +   2.29 +        [fffftt=]  3.39   f ; 35  L
qq  41.9 +   d.19 +   3.29 +        [fffftt=]  3.39   f ; 36  M
qq  42.9 +   d.19 +   4.29 +        [fffftt=]  3.39   f ; 37  N
qq  43.9 +   d.19 +   5.29 +        [fffftt=]  3.39   f ; 38  O
qq  44.9 +   d.19 +   6.29 +        [fffftt=]  3.39   f ; 39  P
qq  45.9 +   d.19 +   7.29 +        [fffftt=]  3.39   f ; 40  Q
qq  46.9 +   d.19 +   8.29 +        [fffftt=]  3.39   f ; 41  R
qq                                                     t ; 42  un
qq  56.9 +   d.19 +  32.29 +        [fffftt=]  3.39   f ; 43  Ø
hv c15         , qq                 [-t----=] 16.29   f ; 44  P CN    none
qq                                                     t ; 45  un
qq                                                     t ; 46  un
qq                                                     t ; 47  un
qq  55.9 +   d.19 +  32.29 +        [fffftt=]  3.39   f ; 48  Æ
qq  29.9 +   d.19 +  17.29 +        [fffftt=]  3.39   f ; 49  A
qq  30.9 + d79.19 +  18.29 +        [fffftt=]  3.39   f ; 50  B
qq  31.9 +   d.19 +  19.29 +        [fffftt=]  3.39   f ; 51  C
qq  32.9 +   d.19 +  20.29 +        [fffftt=]  3.39   f ; 52  D
qq  33.9 +   d.19 +  21.29 +        [fffftt=]  3.39   f ; 53  E
qq  34.9 +   d.19 +  22.29 +        [fffftt=]  3.39   f ; 54  F
qq  35.9 +   d.19 +  23.29 +        [fffftt=]  3.39   f ; 55  G
qq  36.9 +   d.19 +  24.29 +        [fffftt=]  3.39   f ; 56  H
qq  37.9 +   d.19 +  25.29 +        [fffftt=]  3.39   f ; 57  I
hh c11         , qq  d50            [-f----=]         f ; 58  LC      none
qq c23.9 +7d75.19 + c38.29 +        [ffffft=]  1.39   f ; 59  :       d27
hv  c2         , qq                 [-f----=]         f ; 60  UC      none
hv c16         , qq                 [-t----=] 16.29   f ; 61  SUM     none
qq                                                     t ; 62  BLACK
hh b11         , qq                 [-t----=] 16.29   f ; 63  TF      none
hv c10         , qq                 [-f----=]         f ; 64  CR      (b13)

e              ; end compound and tables;
```

[14.4.67]
[Gier algol 4, pass 1, page 35]

[8. END TAPE PASS 1]

```
d i=d74
qq [pass sum]

d e22=k-e14,e47=j        ; Definition of running pass track for next pass.

b k=e23,i=0              ; load pass 1 segment word;
d i=1e21                 ;
qq e2.9+1d74.19-e2.19+1.20+1.24+a31.39 f  ;
e                                         ;

e [final end pass 1]                      ;

s
```

[31.10.66]

[GIER Algol 4, pass 2, page 1]

[Synopsis
Pass 2 recognises byte strings representing identifiers and substitutes a unique byte for each such string. This is done regardless of block structure so that the same identifier will be represented by the same byte throughout the text.
The values of the bytes will be in the range 1023 > <byte> > 511.

Tables.
Pass 2 uses two tables: first letter table[1:28| and table [first free after pass 2: top of store|;
Table holds, starting at top of store, one word (referred to as short word) for each distinct identifier. All words representing identifiers with the same first letter form a chain whose links are the address parts. The end of the chain has address part = 0. The starting point of each chain is given in the addresspart of the corresponding word in the letter table if the first letter is $\leq$ 28 else in the counting part of first letter table [first letter - 28|. A letter which does not start a chain is represented in the letter table by a reference to itself.

The rest of the identifier is assembled as an integer in a base 67 number system.
If this integer < 2↑30 it will be stored in the remaining 30 bits of the short word and bit 40 will be 1.
Otherwise the last 20 bits of the integer will be stored in bit 20 to 39 of short word and bit 40 will be 0. The rest of the integer is stored in one or more words (long words) starting at the first free word at the bottom of table and the address + 1 of the last of these words will be stored in bit 10 to 19 of short word. The first of these words will have bit 0 = 1, the others, if any, will have bit 0 = 0.
The two tables are kept in core after pass 2 for use by the std proc look up at the start of pass 3.

Table look up
Each identifier encountered by pass 2 is added to the table but the linking is not performed.
Pass 2 now compares each entry in the corresponding chain with the new identifier. If the identifier is found in the table the new identifier is removed from the table otherwise the necessary linking is performed.
Finally the address of the short word corresponding to the identifier is put out.

Program:]

```
b k=e22+e14, i=e16-e47, a34,b14,c5,d6; drum block head pass 2;
i=e16                                 ;

d d2=e20-1 ; first identifier value  ; initial i short;

b k=e31, i=0; Load texts
i=e32       ;
d3: tidentifier overflow; used by 1a12
d4: tpattern;           ;  -   - 1a26
e32=i       ;
e           ;

e16: [First letter table starts here. First letter 1-28 uses part 1,
     29-56 uses part 2. From start part 1 and part 2 points at them
     selves (values < 512). First time an identifier with a given first
     letter is encountered the corresponding address in the first letter
     table is changed to point at the short word for that identifier
     thereby starting a new chain (values ≥ 512)]

qq          ; Final i short is stored here for use by the next segment.
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
```

```
                                  ; constants. Used by
c1:  qq  1.19-1.39                ; mask long  5a4
c2:  qq  1.9-1.39 f               ; mask short a5
c3:  m             1              ;            a33
c4:  m           67a              ;            2b3
c5:  m           10               ;            a23
                                  ; start pass 2:
a1:  pmm(e1)   X   1   IZC        ; next: fchar:= next byte; short:= ZB:= t;
     hs  e2              LA       ;   if f char > 512 then
     ga  b1                       ;     begin s:= fchar; goto special actions end;
b1:  bs [f char]Vt 56  NT         ;   if f char > 56 then
     ps (b1)   , hv  a34          ;     begin set return(next - 1);
     ps  a1-1  , hv  e3           ;     Raddr:= fchar; goto output end
b2:  ppn d1 [ilong],ps(b1)        ;   p:= ilong; s:= fchar; R:= 0;


a2:  pm (e1)   X   1              ; next char: swap;
     hs  e2              LA       ;   char:= Raddr:= next byte; finis:= LA:= t;
     ga  b3                       ;   if char > 66 then
b3:  bs [char] t   66  NT         ;     end identifier:
     hvn a4    X                  ;     begin R:= 0; swap; goto ident finis end;
     ck  10    , ml  c4           ;   RM:= M x 67 + byte; finis:= LA:= f;
     hv  a2    X       LZ       ;   if R = 0 then begin swap; goto next char end;
b4:  bsp-510t d2-512[itest]       ; check room: if p-510 > itest then
     ps  a12   , hv  e5           ;     mess(<<no room for identifiers>, 2, 0);
                                  ;
                                  ;   if -, short then store long word:
     gm  p     V       NZA       ;     begin table [p]:= M; p:= p + 1;
                                  ;       swap; goto next char end;
a3:  cl  19    XV      IZA        ; store first long: short := ZC:= f;
     pp  p1    , hv  a2           ;   table[i short] mark ZC:= RM rem 2↑20;
     ck  -19   , ud  a6           ;   RM:= RM : 2↑20; swap;
     hv  a2              LA       ;   if -, finis then goto next char;
                                  ;
                                  ; finis ident:
a4:  pm  512   DXV     NZA        ;   if short then
     ca  0     , hh  a5           ;     begin if Raddr = 0 then goto store short;
     hv  a3    X       LZA       ;       swap; goto store first long end;
     gm  p     , ac (b2)          ;   table[p]:= R; table[ilong]:=
     pp  p1    , it  p            ;     table[ilong] + bit 0; p:= p + 1;
     pt (b6)   , arn c1           ;   set part 2 of (table [i short]) to: (p);
                                  ;   R:= mask long; goto end ident;
a5:  hv  a7    , pm  c2           ; store short: table [ishort] mark ZC:= R;
a6:  gr (b6)   X       MZA       ;   R:= mask short;
       [executed from 2a3]        ;

a7:  pm (e1)   XD -1   IPB        ; end ident: set PB; reset input; swap;
```

[31.10.66]

[GIER Algol 4, pass 2, page 4]

```
[Now an identifier has been read and the situation is as follows:
 s = first char, range 1-56. p = address of first free word after long words.
 store [i short| = short word, part 1 = 0. R ≠ 0 (address of last byte).
 if short identifier: Marks[ishort| = 11, no long words stored.
p = ilong. PB = 1. M = mask short = 2↑30 - 1;
 if long identifier: marks [ishort| = 01 part 2 = p. store [ilong| =
 first long word, bit 0 = 1. store [p-1| = last long word, may be
 the same as first long word. PB = 0. M = mask long = 2↑20 - 1]

                                  ; start look up:
     bs  s-28   , ud se16-28;   if s ≤ 28 then s:= part 1(first letter[s]) else
     ps (se16)          NZ  ;     begin R:= 0; s:= part 2 (first letter[s-28]) end;
     bs  s-511  , hh  a11   ;   if s ≥ 512 then goto look up;
                                ; first identifier in chain: chain:= s;
     gs  b5     , it (b6)   ;   comment address of first letter [f char].
     pt (b5)    V       LZ  ; connect to chain:
a8:  it (b6)    , pa (b5)   ;   if R = 0 then part 2[chain] = i short else
     pm (b6)    DX          ;not found: part 1 [chain]:= i short;
     ps (b6)    V   -1  LT  ;   Raddr:= i short; i short:= s:= i short - 1;
     ps  a12    , hv  e5    ;   if Raddr < 512 then
     it s-512   , pt  b4    ;     mess(«no room for identifiers», 2, 0);
     gp  b2     , ps  a1-1  ;   itest:= s-512; ilong:= p; set return(next);
                                ;   goto output;
a9:  hv  e3     , arn(b5)   ; after long failed: R:= table [chain]
a10: ga  b5     V       LT  ; next test: if Raddr < 512 then goto not found;
                                ;   chain:= Raddr; goto test it;
a11: hv  a8     , gs  b5    ; look up: chain:= s;

b5:  arn -1 [chain]     IPA ; testit: R:= set PA (table[chain]);
b6:  cm d2[ishort], hva10   ;   if R masked M ≠ table[ishort] masked M
                                ;     then goto next test;
     hv  a14            LPC ;   if LPC then goto found;
     gp  b7     V       NPC ;   if -, NPC goto next test;

d a12=i-1 [mess descr]      ; test rest of long: ilong new:= p;
     hv  a10    , qqn d3    ;   i long old:= part 2 (R);
     ck  10     , ga  b8    ;
a13:                        ; test next long: i long new:= i long new - 1
b7:  arn[ilong new]t-1 ITA  ;   i long old:= i long old - 1;
b8:  sr [i long old]t-1     ;   if table [ilong new] ≠ table [ilongold] then
     hh  a9             NZ  ;     goto after long failed;
     hv  a13            NTA ;   if bit 0 (table[i long new]) = 0 then
                                ;     goto test next long;
a14: pm (b5)    DX          ; found: Raddr:= chain; set return (next)
     ps  a1-1   , hv  e3    ;   goto output;
```

```
[Patterns:
 This logic uses four states (as an address in s). The state table
 contains for each state four jump instructions packed in two
 words. Each jump instruction corresponds to an input byte
 class. The comments below describe for each state the new
 state and the action for each of the four input byte classes in
 the following order:                digit     lined digit
                                     letter m  other]

a15: pan b12   , ud  a21   ; pattern: pos:= word:= M:= R:= 0;
     grn(b6)   , hv  a24   ;   reset input; goto end term;

a16: arn(e1)   t   1   IOA ; get next: digit := next byte;
     hs  e2              LA ;   copy:= f;
     ga  b13   , ca  13    ;   if digit = letter m then
     srn c3    , hv  s1    ;     goto m action [state];
     pp (b13)  , it  501   ;   if digit is normal digit then
     bs  p445  , hv  s     ;     goto digit action [state];
     bs  p512  t   501     ;   if digit is lined digit then
     arn b11   , hh  s     ;     goto lined digit action [state];
                           ;   goto other action [state];
a17h:hh  s1    , ns (b11)  ; mval: R:= 2↑bits - 1 shift 40 - bits;
     tk  s40   , hv  a23   ;   goto acc;

a18: tk  2     , ar  b11   ; mult bits:
     tk  1     , ga  b11   ;   bits:= bits × 10 + lined digit value;
     it  p10   , pp (b11)  ;   if bits > pos + 40 then state:= 0;
     it (b12)  , bs  p-40  ;   goto get next;
                           ; test: if R ≠ 0 then state:= 0; goto next;
a19: hs  a16           NZ  ;           state 0:    0, get next      0, get next
     hv  a16   , hs  e5    ;           in error    0, get next      alarm exit

a20: pm (b6)   , qq  sd4   ; alarm exit: mess («<pattern», 0, 1);
a21: qqn(e1)   Xt   -1     ; exit: reset; R:= word; goto output R;
     [exc. by a15, a22]
a22h:hv  a31   , ud  a21   ; reset acc: reset; swap; M:= 0;
b11: ns [bits] , cl  s40   ;   RM:= R long shift 40 - bits;
     [b11 has 00 in bits 10-11]
a23:                       ; acc: word:= word + R shift pos;
b12: ck [pos]  , ac (b6)   ;   pos:= pos - bits;
     nt (b11)  , qq (b12)  ;   comment M ≠ 0 indicates now too big value;
                           ; end term: R:= M; M:= bits:= 0;
a24: pan b11   X           ;   state:= 1; goto test;

a25: hs  a19   , hh  a26   ;           state 1:    0, get next      2, mult bits
     hv  a19   , hv  a20   ;           after term  0, get next      exit

a26: hv  a27   , hs  a18   ;           state 2:    3, mult val      2, mult bits
     hh  a17   , hh  1a19  ;           in bits     1, mval, acc     alarm exit

a27: hs  a28   , hh  a22   ;           state 3:    3, mult val      1, reset acc
     hv  a19   , hh  a22   ;           in value    0, get next      1, reset acc
a28:
b13: arn[digit]Dt  -57     ; mult val: RM:= M × 10 + digit value;
     ck  10    , ml  c5    ;   goto test;

a29h:hv  a19   , ns (b6)   ; end pass 2:
     it  sd2   , pa  2e4   ;   inf 1:= initial ishort - ishort;
     it (b6)   , pa  e16   ;   save final i short for use by next segment;
     nt  d1    , it (b2)   ;   inf 2:= ilong - initial i long;
     pa  3e4   , hhn e29   ;   R:= 0; goto next segm;
```

```
[output four outputs 4 bytes, either from R (copy = f) or from
 input (copy = t).]
a30: arn(e1)   t   1   LZA ; output four:
     hs  e2            LA  ;   if copy then Raddr:= next byte;
     ga  b14   , ck  10    ;   out byte:= Raddr; R:= R shift 10;
b14: pm  -1    DX          ;   output (outbyte);
     hs  e3                ;   i:= i + 1 mod 4;
     ncn 0[i]  X  -256     ;   if i ≠ 0 then
a31: pm  r-1   , hv  a30   ; output R: begin LA:= f; goto output four end
                           ;   goto next;
a32: hv  a1    , ps  1     ; beg code:
     arn(e1)   t   1       ;   for Raddr:= next byte while Raddr < 512 do
     hs  e2            LA  ;     output (Raddr);
     hv  e3            NT  ;
     tk  -30   , sc  e4    ;   CRcount:= CRcount + R × 2↑(-30);
     tk  30    , ps  a1-1  ;   set return (next); goto output;
a33: hv  e3    , arn c3    ; CR: CR count:= CRcount + 1;
     ac  e4    , hv  a1    ;   goto next;

a34: bs s-1014 , ps  1014  ; special actions: if s > 1014 then s:= 1014;
     arn(sd5)  D           ;   output part 1 (act table[s - 1008]);
     ps  sd6   , hv  e3    ;   goto act part (act table[s - 1008]);


[Action table: used for action bytes > 511: part 1 is unconditionally
 output and the action specified in right half entered
                             input. action.]
     qq  264   , hh  a32   ; 1008 begcode
     qq  283   , hh  a29   ; 1009 end pass
     qq  77    , hh  a33   ; 1010 CR
     qq  78    , hv  a30   ; 1011 short str
     qq  79    , hv  a30   ; 1012 long str
     qq  80    , hv  a30   ; 1013 layout
     qq  80    , hv  a15   ; 1014 - 1023 pattern

d d5=i+9, d6=d5-1          ; define act table bases.
d1:  qq [pass sum]         ; define initial i long = first free after pass 2;

d e22=k-e14, e47=j; Define load parameters.
b k=e23, i=0       ; Load segment word 2.
i=2e21             ;
qq e16.9+1d1.19-e16.19+2.24+a1.39;
e                              ;

e [end pass 2]     ;

s
```

[27.4.67]
[Gier Algol 4, pass 3 phase 1, std proc, page 1]

[This segment assumes that the first letter table and the table
from pass 2 still is in core.
It takes in thetable of standard procedure identifiers word by word
using the word driver in GP. Each identifier is looked up in
the pass 2 table and if found 2 bytes are output:
1. The running number (T) in the standard procedure table.
2. The pass 2 identifier.

Format of a standard procedure identifier:
1. A short word corresponding to short word in pass 2 with:
part 1 = first letter (range 1-56).
If long identifier then part 2 = 0 and bit 40 = 0
otherwise bits 10-39 holds the rest of the identifier and
bit 40 = 1.
2. If long identifier:
One or more long words as in pass 2 but in such an order
that the one with bit(0) = 1 comes last i.e. opposite
the order in which pass 2 generates them.

The byte pair output (see above) is preceded by a byte 0
(to stop on in pass 4) and followed by <first free ident>
which is stored by pass 2 in part 1 of letter table[0].

This segment is loaded at 29e16, which is just after
the first letter table in pass 2.]

```
b k=e22+e14, i=29e16-e47, a16, b8, c3, d7; drumblock head
i=29e16                                  ;

b k=e31, i=0  ; load texts
i=e32         ;
d1: tstd proc format;
d2: tzero;    ;
d7: tdouble std proc;
e32=i         ;
e             ;
```

```
a:   hs  e5                  ; zero mess: mess (‡<zero‡, 2, 3);
     hh  r1    , qqn pd2     ;   pause;
     qq        , ud  14e4    ;
     lyn r-1   , ud  16e4    ;
a1:  arnf b7   , srf b7      ; start: if wrong floating point zero then
     grf  b8                 ;
     ncn(b8)   , hv  a       ;   go to zero mess;
     hsn e3                  ;   output (0);
     qq  r2    , arn b1      ;   word driver to core d4ff;
     ps  r     , hh  e29     ;   words:=bits(10,19,M)-1;
     cln -30   , ac  b6      ;

a2:  pp  0     , hs  a15     ; next proc: p:=0; next word;
     gr  b7    , ga  b1      ;   short word:= R; first letter:= s:= Raddr;
b1:  ps d4 [first letter]    ;   if s > 56 ∨ s < 1 then goto proc err;
       [see 1a1]
     bs  s-513 t  -457       ;
     hv  a6                  ;   if -,mark A then
     hv  a4            LA    ;     begin take long: p:= 1; loop long:
     pp  1                   ;     long word[p]:= R:= next word;
a3:  hs  a15                 ;     if p > 5 then goto proc err;
     bs  p-5   , hv  a6      ;     p:= p + 1;
     gr  pb7   , pp  p1      ;     if R ≥ 0 then goto loop long;
     hv  a3            NT    ;     M:= mask long; PB:= f,
     pm  c1    V       IPB   ;     end
a4:  pm  c2            IPB   ;   else begin M:= mask short; PB:= t end;
                             ; start look up: s:=
     bs  s-28  , ud se16-28;    if s ≤ 28 then part 1 (first letter table[s]);
     ps (se16)         NZ    ;   else part 2 (first letter table[s]);
     bs  s-511 , hh  a9      ;   if s ≥ 512 then goto look up;
a5:  arn(b5)   D   1         ; not found: T:= T + 1;
     hv  a2            NT    ;   if T < 512 then goto next proc;
a6:  hs  e5                  ; proc err:
     hv  a13   , qq  sd1     ;   mess (‡<std proc format‡, 0, 1); goto skip;
a7:  arn(b2)                 ; after long failed: R:= table[chain];
a8:  ga  b2    V       LT    ; next test: if Raddr < 512 then goto not found;
                             ;   chain:= Raddr; goto test it;
a9:  hv  a5    , gs  b2      ; look up: chain:= s;
b2:  arn [chain]       IPA   ; test it: R:= set PA(table[chain];
     cm  b7    , hv  a8      ;   if R ∧ M ≠ short word ∧ M
                             ;     then goto next test;
     hv  a11           LPC   ;   if LPC then goto found;
     pa  b3    V b7    NPC   ;   if -,NPC then goto next test;
     hv  a8                  ;   i1:= 0;
     ck  10    , ga  b4      ;   i2:= part 2 (R);
a10:                         ; test rest of long: i1:= i1 + 1; i2:= i2 - 1;
b3:  arn[i1]   t  +1   ITA   ;   if table [i2] ≠ long word [i1] then
b4:  sr [i2]   t  -1         ;     goto after long failed;
     hv  a7            NZ    ;   if bit 0 (long word[i]) = 0 then
     hv  a10           NTA   ;     goto test rest of long;

a11: pm (b2)           IPA   ; found: if markB(table[chain]) then
b5:  arn 0 [T] DVt 1   NB    ;   mess(‡<double std proc‡,0,1)
     ps  a12   , hv e5       ;   else
     pm (b2)   D       M     ;     begin T:= T+1;  output(T);
     hs  e3            IPB   ;     output(chain);  set mark B on (table[chain])
     acn(b2)   X       MPC   ;     end;
     ps  a2-1  , hv  e3      ;   goto next proc;
a12=i-1
     hv  a5    , qq  sd7     ;   mess descr for double std.proc;
```

```
a13:  pp  0     , ps  1      ; skip: p:=0; set return(next word);

a15:  hsn d5            IZC ; next word: sum ok:=transport ok:=true;
      ar  2     D       LA  ;   Get word; proc sum:=procsum+marksR+R;
      ar  1     D       LB  ;   restore R;
      ac  b8    , arn e16-1 ;   words:=words-1;
b6:   bt -1[words] Vt-1LZA  ;   if -,transport ok then
      hs  e5                ;    mess(‹<pass medium›,2,0);
      hr  s1    , qqn e46   ;   if words≥0 then return;
                            ;
      bs  p     , hv  a6    ; finis: if p>0 then goto proc err;
      arn e16   , hs  e3    ;   output(final i short);
      arn 8e4   , hs  e11   ;   restore inbuf2;
      lk 42e13  , arn b8    ;   if proc sum≠0 then
      hs  e5            NZ  ;    mess(‹<pass sum›,2,0);
      hhn e29   , qqn e45   ;   R:=0; goto next segm;

b7:   qq 1 [short word] t256;   Also used for test of zero arithmetic.
      [see a1-1]
      qq [long words 1:5]   ;
      qq  ;
      qq  ;
      qq  ;
      qq  ;
b8:   qq [proc sum]         ;


c1:   qq 1.19-1.39    ; mask long
c2:   qq 1.9-1.39f    ; mask short
c3:   m          1    ;

d4:   qq [pass sum]         ;   word driver is taken in to d4.
d5:   [entry to word driver];

d e22=k-e14, e47=j          ;   define load parameters
b k=e23, i=0                ;   set segment word 3
i=3e21                      ;
 qq a.9+d5.19-a.19+3.24+a1.39;
e                             ;

e [final end]               ;
s
```

```
d e49=3        ; tape number := 3;

[After i follows STOPCODE, SUMCODE and a sum character]
ia T2
s
```

[27.11.67

T3 Gier algol 4
10]

[Here follows STOPCODE and CLEARCODE]

```
<e49-2, <-e49+4, x  ;   test tape number
1 wrong tape
s                   ;
>                   ;

d e53=10            ;   version number

<e53-e50,           ;   if version number T3 gr max version number
                    ;   then the following definitions are loaded;
d e61=1             ;   define version number T1 and L1
d e62=9             ;   define version number T2
d e63=e53           ;   define version number T3
d e64=4             ;   define version number T4
d e65=5             ;   define version number T5
d e66=6             ;   define version number T6 and L2
d e67=7             ;   define version number T7 and L3
d e68=8             ;   define version number T8 and L4
d e50=e53
>                   ;
```

[3.8.66]

[GIER ALGOL 4, Pass 3, page 1]

```
b k=e22+e14,i=e16-e47, a30, b26, c87, d60;
i=e16                                                                    ;
d d4=e20 [max stack]                                                     ;

[Input byte values]
d d17=167[;], d18=184[(]                                                 ;
d d7=241 [trouble]                                                       ;

[Output byte values]
d d5=39[beg func]                                                                    ;
d d9=134[end else exp], d10=128[proc;], d11=2[litreal]                               ;
d d12=1[lit integer], d15=22[do]                                                     ;
d d16=e20[dummy identifier]                                                          ;
d=5[value-non value spec], d13=51[undeclared]                                        ;
d33=68[unspec], d35=18[end proc no type]                                             ;
d36=46[decl switch], d43=e20[switch dummy ident]                                     ;
d44=69[specgen], d45=30[end spec]                                                    ;
d46=19[end type proc], d47=52[spec integer]                                          ;
d48=166[case exp], d49=133[delete call]                                              ;
d37=57[spec value int]                                                               ;

[Stack representations]
d d3=12[beg block], d6=32[func], d8=1[else exp]                          ;
d d19=15[beg proc], d34=1[do-single do]                                  ;

b:   qq 1.3-1.39                        ; mask. c10,
[1b] qq t 256                           ; 1.0 floating, c36, c40
[2b] qq 3 t 320                         ; 10 floating. c38, c39, c41, c56
m                                       ;
[3b] qq 10.39                           ; c56,
[4b] qq 57                              ; 57.c39,
                                        ; c43,
[5b] qq 1.39                            ; 1
[6b] hs c7                              ; instruction, c7,
[7b] qq 1020.9+409.19+614.29+410.39     ; 0.1 floating
[8b] qq 1023.39                         ; 1023, 4a9
b1:  qq                                 ; decl.c13, c15, c16, c17, c20
[1b1]qq                                 ; N. c60, c36, c39, c40, c48, a19, a9, c67, c59,
                                        ; c56,
[2b1]qq                                 ; factor. c36, c38, c48
                                        ; 1c52, 1c53, c41, c48, a16
[3b1]qq                                 ; x. c39,

d d40=1                                 ; If redefined, d40=0: output of
s                                       ; state, KA, KB=10, or stack = 00
```

```
b k=e31, i=0                                ;
d i=e32                                     ; Alarm texts, used in:
d51: t-delimiter;                           ;    1c5,
d42: toperand;                              ;   c10-1, 2a12
d e32=i                                     ;
e                                           ;

c2: pi   0        V   -5   LQB   ; AFTER TROUBLE: if first after trouble then
    pi   0        t   -9         ;        first after trouble:= false else
    hv   c3                      ;        introuble:= false; go to NEXT 1;
c66:hs   e3       ,   ps  i      ; OUT: output(par M);
c1: pa  [operand] DV      NQA    ; NEXT:  operand:= 0;
    pa   c1       ,   hv  c2     ;    ifintrouble then go to AFTER TROUBLE;
c3: pmn (e1)      X   1          ; NEXT 1: byte:= input;
    hs   e2                LA    ;    if byte < 512 then go to NOT OPERAND;
    hv   c5                NT    ;    if -, introuble then
a6: hs   e3[or a5]         NQA   ;    begin if -, output identifier then
   [output identifier: true=e3, false=a5]; keep else output(byte) end;
    pa   c1       t   1    NQA           ; if -, introuble then operand := 1;
c6: pmn (e1)      X   1                  ; AFTER OPERAND: byte:= input;
    hs   e2                LA    ;    if byte ≥ 512 then alarm(d41);
c5: ga   a1       V            NT  ; NOT OPERAND: R:= table[byte];
    hs   c7       ,   qq  s+d51  ;    marks:= marksof(table[byte]);
    hv   i+5                LKB   ;    skip output for KB = 1
    hs   e7                 LKA   ; Special state output: delimiter
    sy   27                 LKA   ;          comma
    sy  (c1)                LKA   ;          operand
    sy  (i+3)               LKA   ;          state
d i=i-d40-d40-d40-d40-d40           ; remove special output unless d40=0
a1: pmn [byte]    Xt  d2            ;    if marks > 1 then go to SPECIAL;
    hv   c9                 LA    ;    OA:= bit(state, R); if marks = 0 then
a2: ck   27 [state]         IOA   ;    begin byte:= byte - 1;
    hv   a3                 LB    ;    R:= table[byte]; marks:= marksof(table[byte]);
    pmn (a1)      X   -1            ;    OB:= bit(state, R);
    ck  (a2)                IOB   ;    if marks  = 0 then
    hv   a4                 LB    ;        begin byte:= byte - 1;
    pmn (a1)      X   -1            ;        R:= table[byte];
    ck  (a2)                      ;        if OB then byte := byte - 4
    qq  (a1)      t   -4    LOB   ;        end;
a4: qq  (a1)      t   -2    LOA   ;    if OA then byte:= byte - 2
a3: pmn (a1)      XV  -1    LO    ;    end; if bit(state,R) then byte:=byte-1;
a22:pmn (a1)      X               ;CONTROL WORD FOUND:
    ck  (c1)      V         NC    ;    R:= table[byte]; marks:= marksof(table[byte]);
    hv   c8                       ;    if marks > 0 then go to SEARCH;
    hv   c10                LQA   ;    if -, bit(operand, allowed operand part(R))
    pmn (a1)      XV        LO    ;        ∧ -, introuble then alarm (d42);
    hs   c7       ,   qq  s+d42   ;
c10:mb   b        ,   ga  a2      ; NORMALACTION: state:= newstatepart(R);
c9: hv   c35                LC    ; SPECIAL: if marks = 3 then go to INITIALIZE NUMBER;
    gt   a        ,   cl  -9      ;    parM:= outpart(R); parR:= stackpart(R);
a:  ck   -11, hv [switching part];    go to instruction[switchingpart(R)];
a5: ga   b6       ,   hr  s1      ;    procedure keep; ident:= byte;
b6: ca [ident]-1  ,   hs  s1      ;    ident word
```

```
b k=e31, i=0                          ;
d i=e32                               ; Alarm texts, used in:
d53: tdelimiter;                      ;    a14-5, 2c29
d54: t-operand;                       ;    a14
d55: ttermination;                    ;    1a30
d52: thead;                           ;    1b4
d e32=i                               ;
e                                     ;

c63:pi   0        t   -13             ; START PASS 3: introuble:= first after trouble:=
    pm   a20      V                   ;        stackidentifier:= false;
a20:hv   c51      ,   hv  c51         ;    store[0]:= jump to error 2;
    gm   0                MA          ;    for i:= upper stack limit step -1 until 0 do
a21:grn  d4       t   -1  M           ;        store with marks(stack[i], 0, 0);
    bs  (a21)     t   d1              ;    ds:= 0; state:= 27;
    hv   a21                          ;    go to NEXT;
    pp   d1       ,   hh  c66         ;
c62:hhn  e29                          ; END PASS 3: R:= 0; goto new segm;
c7: arn  s        ,   ps  c66         ; ALARM: procedure alarm(n); value n; integer n;
    hv   c6                 LQA       ;    begin if introuble then go to AFTER OPERAND;
    sr   6b       ,   ac  a10         ;    errormessage(n);
    hs   e5                           ;    byte address:= byte address - 1;
a10:qq   d7       ,   qq              ;    byte:= trouble;
    pa   c1       ,   grn a10         ;    introuble:= first after trouble:= true;
    pa   a10      t   d7              ;    operand:= 0; go to NOT OPERAND
    qq  (e1)      t    -1             ;    end;
    pi   12       t    -13            ;    comment errormessages: 34: stack, 51: -delimi-
    arn  a10      ,   hv  c5          ;    ter, 52: head, 42: operand, 53: delimiter, 54:
                                      ;    -operand, 55 termination, 56: number
c8: sy   64                 NKC       ; Special output, stack: CARRET
    arn (a1)       D        NKC       ;
    hs   e7                 NKC       ;                 byte
    sy   27                 NKC       ;                 comma
    sy  (c1)                NKC       ;                 operand
    pmn (a1)      X                   ; reset A and M
d i=i-d40-d40-d40-d40-d40-d40         ; remove special output
    gr  (a30)     DV        LA        ; SEARCH: if marks = 1 then alarm(d43);
    hs   c7       ,   qq  s+d53       ;    comment delimiter;
    ck   10       ,   gt  a11         ;    lim 1:= part 1(R); lim 2:= part 2(R);
                                      ;    base:= part 3(R);
    pm  (c1)      DX                  ;    if marks = 3 then go to a12;
    hv   a12                LC        ;    if operand = 0 then alarm(54);
    pm   d8       DVX       NZ        ;    comment - operand;
a14:hs   c7       ,   qq  s+d54       ;    if stack[ds] ≠ elseex then go to a13;
    nc  (p)       ,   hv  a13         ;
    sy  (p)                 NKC       ; Special output: top of stack
d i = i - d40                         ; remove special output unless d40 = 0
    pm   d9       DV                  ;    R:= end else expr;
```

```
c55:cl   -9                          ; c55:
    hs   e3        X                 ;    output(R);
c54:pp   p-1                         ; c54: ds:= ds - 1;
a13:sy   (p)                  NKC    ; Special output: top of stack
d i = i - d40                        ; remove special output unless d40 = 0
    is   (p)                         ; a13: st:= stack[ds];
a30:bs   s0        t   0             ;    if st+lim 1 < -512 ∨ st > lim 2-lim 1 then
    hs   c7        ,   qq  s+d55     ;    alarm(d55); comment termination;
a11:is   (p)       ,   pmn s[base];
    hv   c10       X       NC        ;    if marks = 0 then go to NORMAL ACTION;
    hv   c55       X       LA        ;    if marks = 1 then begin R:= part 4(R);
    hv   c54       X                 ;    go to c55 end; go to c54;
a12:hv   a13                 LZ      ; a12: if operand = 0 then go to a13;
    nc   1                           ;    if operand ≠ 1 then
    hs   c7        ,   qq  s+d42     ;    alarm(d42);
    pm   d10       DX                ;
    hs   e3                          ;    output(proc;);
    hv   a13                         ;    go to a13;
c11:pt   b26       t   d35           ; SETBLOCKPROC: end proc:= no type end proc
b2: pa   a6        t   a5            ;    output identifier:= false;
    pa   b4        ,   ud  b5        ;    head alarm:= false; ident:= dummy;
c12:ca   (p)                         ; SET BLOCK: if stack[ds] = parR then
    pa   p         t   d3            ;                 stack[ds]:= 'begblock';
c13:gm   b1        ,   hv  c1        ; SET DECL: decl:= parM; go to NEXT;
c14:pt   b26       t   d46           ; ADD DECL PROC: end proc:= type end proc;
b5: pa   b6        t   d16           ;    ident:= dummy ; head alarm:=
    pa   b4        ,   ud  b2        ;    false; output identifier:= false;
c15:xr             ,   ac  b1        ; ADD DECL: decl:= decl + parM; go to NEXT;
    hv   c1                          ;
c17:pm   b1                          ; DECL ENT: parM:= decl;
c18:pp   p1        ,   bs  p-d4      ; ENT OUT: ds:= ds + 1;
    hs   c7        ,   qq  e34       ;    if ds > maxds then alarm(38); comment stack;
c19:ga   p         VX                ; CH OUT: stack[ds] := parR; go to OUT;
c20:pm   b1        X                 ; DECL: parM:= decl; go to OUT;
    hv   e3                          ;
c21:pp   p1        ,   bs  p-d4      ; ENT: ds:= ds + 1; if ds > maxds then
    hs   c7        ,   qq  e34       ;        alarm(38); comment stack;
c70:ga   p         ,   hv  c1        ; CH: stack[ds]:= parR; go to NEXT;
c28:hs   ne3       X                 ; TROUBLEPROCEND: output (par M);
c22:arn  p         ,   cl  -9        ; PROCEND: R:= stack[ds];
    tk   19                          ;    par M:= par t4(R);
    hs   e3                          ;    output(part 2(R));
c61:qq   (2e4)     t   1             ; BLOCK COUNT: information 1:= information 1 + 1;
c23:pp   p-1                         ; AN OUT: ds:= ds - 1; go to OUT;
c4: hv   e3        X                 ;
c24:pp   p-1       ,   hv  c1        ; AN: ds:= ds - 1; go to NEXT;
c64:hs   e3        X                 ; DO: output(par M);
    arn  d15       DX                ;    parM:= 'do';
    bs   (c34)     ,   ar  r1        ;    if count ≠ 0 then par R:=
    qq   d34       ,   hv  c19       ;    par R + do difference; goto CHOUT;
c26:bs   (c1)                        ; LEFT PARENT: if operand > 0 then
    pmn  d5        D                 ;        begin parR:= 'func'; parM'begcall' end;
    arn  d6        D       LZ        ;    go to ENT OUT;
    hv   c18                         ;
```

```
c68:hs   e3       X                ; RIGHT CALL: output(par M);
    arn  d49      DX               ;    par M:= delete call;
c25:pp   p-1      ,   ga  c1       ; RIGHT: ds:= ds - 1; operand:= parR;
    hsn  e3       X                ;    output(parM); go to AFTER OPERAND;
    hv   c6                        ;
c69:                               ; BOUNDS:
    mb   1023     D                ;    R:= part 1(R);
    hs   e3       X                ;   output(par M); M:=R;
    hvn  c25                       ;    R:= 0; goto RIGHT;
c27:ga   b22      ,   can(c1)      ; PLUSMINUS: monadic:= par R;
b22:pm [monadic]-1 D               ;    if operand = 0 then par M:= monadic;
    hv   e3       X                ;    goto OUT;
c29:bs  (c1)                       ; BINARY: if operand = 0 ∨ state > 7 ∨ introuble
    bs  (a2)      t   7    NQA     ;       then alarm(d43);
    hs   c7       ,   qq  s+d53    ;    comment delimiter;
    ga   a2       X                ;    state:= parR; go to OUT;
    hv   e3                        ;
c30:pi   8        t   -9           ; AN TROUBLE: introuble := true; ds:= ds - 1;
    pp   p-1      ,   hv  c6       ;    go to AFTER OPERAND;
c32:arn  5b       ,   ac  e4       ; CR: CRcounter:= CRcounter + 1;
    hs   e3       X                ;    output(parM);
    bs  (c1)           NQA         ;    go to if operand = 0 then NEXT 1
    hv   c6                        ;         else AFTER OPERAND;
    hv   c1                        ;
b8: qq 1.19+50.35+1.39-1.36        ; instruction modifier (3c16)
c16:pp  p+1       ,   gp  b7       ; FORMALLIST: ds:= stack entry:=
    arn b1        ,   ck  -10      ;     ds+1; stack[ds]:= instruction
    ar  b6        ,   sr  b8       ;
b7: gr  -1 [stack entry]   MA      ;   (ca <ident>, qqn decl) ; cf. 1b26
    gp  b17       ,   hh  c71      ;   goto SET STOP;
c72:arn   b6      ,   hs  (b7)     ; FORMAL: R:= ident word; call
c47:                               ;   (store [stack entry]);
[1] gp   b4       ,   hh  c66      ; return 1: head alarm:= true;
                                   ;   goto NEXT;
b9: qq            ,   hs  s3       ;    stop instruction, 1c71
[3] hv   c47           LZ          ; return 3: if R = 0 then goto return 1;
                                   ;    stack [ds]:= R;

c71:gr   p        ,   pp  p1       ; SET STOP: ds:= ds + 1; if
    pm   b9       ,   bs  p-d4     ;    ds > limit then ALARM (38);
    hs   c7       ,   qq  e34      ;    stack[ds]:= instruction(qq,
    gm   p             MA          ;    hs s3); goto NEXT;
    hh   c66                       ;
```

[When we enter FORMAL the top of the stack has the following form:

```
     stack entry: ca procedure identifier, qqn decl
                  ca formal ident 1      , hs  s1
                  ca formal ident 2      , hs  s1
                  . . . . . .
               p : qq                    , hs  s3
```

No two identifiers are the same]

```
c46:pt   b10      V   b11        ; VALUE: mod:= m2; goto SETSPEC;

c74:pt   b10      t   b13        ; FIRST SPEC: mod:= m1;
    pt   b13                     ; SETSPEC: Store [m1]:= 0;

c75:ga   b12      X              ; SECOND SPEC: value
    ck   -10      ,   ac  b13    ;    allowed:= Rpar; store[m1]:=
    hv   c1                      ;    store[m1] + M par; goto NEXT;

[The current form of the top part of the stack:
     stack entry:  ca procedure identifier, qq n decl
For each formal identifier, one word in one of three formats:
(1) No value, no spec. before     ca formal identifier , hs  s1
(2) Value, no spec.               ca formal identifier , hss c73 - c76
(3) Already specified             ca formal identifier , qqn specif.
At end of list:                   qq                   , hs  s3
No two identifiers in the list are the same]

c76:arn  b6       ,   hs (b7)    ; SPEC COMMA: R:= ident word;
b10:                             ;    Call (store [stack entry]);
[1] qqn           ,   ar  [mod]  ; return 1: R:= 0;
                                 ; modify: stack [s]:= stack [s] +
[2] ac   s        ,   hh  c66    ;    R + store[mod]; goto NEXT;
[3] hv   c47                LZ   ;    if R = 0 then begin head alarm:= true;
                                 ;    goto NEXT end;
    gp   b4       ,   ar  b13    ;    head alarm:= true; R:= R + mod 1;
    hv   c71                     ;    goto return 3;
c73:gr   s        ,   arn b14    ;    stack[s]:= R; R:= value modifier;
b12:nc   0        ,   gpn b4     ;    if value no then begin R:= 0; head alarm:=
   [value allowed: yes = d, no=0];    true end;
    tk   -10      ,   hh  b10    ;    goto modify;
b14:qq   d                       ;
b26: qq[no.9+end.19+] d45.29+d19.39;
[1b26] gr 0       t   5  MA      ;
c77:ga   b16      ,   srn  b7    ; COMPL HEAD: no spec:= par R;
    ar   p-1      D      IQA     ;    R:= ds - 1 - stack entry;
    ar   1b26     ,   ga  b26    ;    no of par:= R;
    ca   0        ,   ac (b7)    ;    if R = 0 then add 5 to (
                                 ;      part 2 [stack entry]);
    ncn (b4)                     ;    introuble:= false; if head
    pt  (b7)      t   d13        ;      alarm then part 2 [stack entry]:=
                                 ;      undeclared;
b17:arn [i]       ,   ga  b18    ;    for i:= stack entry step 1 until ds-1 do
    ck   -10                IOA  ;    begin R:= stack[i]; id:= part 1(R);
    ck   20                      ;    OA:= op2(R) ≠ qqn; R:= part 2(R);
b16:arn [no spec] D         LOA  ;    if -, OA then R:= no spec;
    ga   b19      ,   ca  d33    ;    spec:= R;
    gp   b4                      ;    if R = unspec then head alarm:= true;
b18:pmn  -2 [id]  DX             ;    output(id);
    hs   e3                      ;
b19:pmn  -4 [spec]DX             ;   output(spec);
    hs   e3                      ;
    arn (b17)     Dt   1         ;
```

```
    nc   p        ,   hv  b17    ;    end for i;
    pan  a6       X   e3         ;    output identifier:= true;
    arn  b26      ,   ck  -10    ;    The top word of the stack
    cl   -20      ,   ck  10     ;    is set to qq beginproc.9+
    cl   -29      ,   pp  (b7)   ;    no of param.19 + endproc.39;
    gm   p        ,   hs  e3     ;    output (end spec);
    qq   (e1)     t   -1         ;    byte address:= byte address - 1;
b4: ncn[head alarm],  hs  e5     ;    if head alarm then
    ps   c66      ,   qq  s+d52  ;    ALARM(‹<head›);
    pm   (b6)     D              ;    if operand = 1 then
    arn  c1       ,   ca  1      ;    output(ident);
    hs   e3       X              ;
    hv   c6                      ;    goto AFTER OPERAND

c78:ga   b16      ,   nsn(a6)    ; SEMICOLON: code:= par R; if -, output
    ca   sa5      ,   hh  c77    ;    identifier then goto COMPL HEAD;
    pa   a1       t   b25        ;    byte:= b25;
    hv   a22                     ;    goto CONTROL WORD FOUND;
c31:pa   c34      ,   hv  c4     ; FOR: for count:= 0; goto OUT;

c36:it   -1                      ; WHILE COUNT: count:= count - 1;
c34:qq [for comma count] t 1     ; COUNT CHOUT: count:=
    hv   c19                     ;    count + 1; goto CHOUT

c82:ck   10       ,   tk  30     ; CODE: output(par R);
    hs   e3       ,   ps  i      ;
    arn (e1)      t   1          ; Q: R:= input;
    hs   e2                LA    ;    if R ≥ 0 then
    hv   e3                NT    ;    begin output (R); goto Q end;
    tk   -30      ,   sc  e4     ;    CR counter:= CR counter
    tk   30       ,   hs  e3     ;    + R; output (R);
    ps   c66      ,   hv  c4     ;    goto OUT;
b23:qq 1.19+d35.39               ;    begin label proc, stack word
c83:ca  (p)                      ; SWITCH: if stack [ds] = par R then
    pa   p        t   d3         ;    stack [ds]:= 'beg block';
    arn  b23      ,   gr  p1     ;    stack[ds + 1]:= begin label proc;
    hv   c21      X              ;    goto ENT;

c84:                             ; SWITCH ASSIGN:
    qq   d36      ,   hs  a29    ;    output(decl label proc with par);
    qq   d43      ,   hs  a29    ;    output(i);
    qq   d37      ,   hs  a29    ;    output(spec value integer);
    qq   d45      ,   hs  a29    ;    output(end spec);
    qq   d48      ,   hs  a29    ;    output(case expr);
    qq   d43      ,   hs  a29    ;    output(i)
    arn  17       D              ;    par R:= :=switch
    ps   c66      ,   hv  c18    ;    goto ENTOUT;
a29:arn (s)       D              ;
    hv   e3                      ;
```

```
c37:pmn (e1)      X   1          ; NEXT OF NUMBER: s:= input(byte);
    hs   e2                LA    ;    s:= if 56 < s ∧ s < 67 then 57
    ga   r1                      ;        else if 66 < s ∧ s < 77 then s - 9
    ps                           ;        else 56;
    bs   s445      t   501       ;    go to instruction[numbers +
    ps   57        ,   hh  r3    ;        bits(numberstate, numberstate + 4,
    bs   s435      t   501       ;        numberstatetable[s])];
    ps   s-9       ,   hh  r1    ;
    ps   56        ,   arn sd2   ;
a15:ck [numberstate], tk  -5     ;
    ga   r1                      ;
    hv [switchingpart]t  c38     ;
c58:srn  57        D             ;    procedure mult 10;
    ar   (e1)      ,   ck  10    ;    MR:= MR × 10 +
    ml   3b        ,   hv  s1    ;        symbol - 57;
c60:gm   (a17)     D             ; LOGIC VALUE: kind:= outpart;
    cl   9         ,   gm  2b1   ;    n2:= bit 8 (par R);
    tk   1         ,   tk  -40   ;    n1:= bit 9 (par R) from 0 to 39;
    hh   a8                      ;    goto a8;
c33:gm   (a17)     D             ; LITERAL: kind:= outpart;
    pa   a7        t   3         ;   for i:= 0 step 10 until 30 do
    arn  (e1)      t   1         ;    pack(M, i, i+9, input);
    hs   e2                LA    ;
a7: bt   0         t   -1        ;
    cl   10        ,   hv  r-3   ;
a8: cl   -30       ,   gr  1b1   ; a8: n1:= M;
    hv   c67                     ;    goto OUTPUT LITERAL;
a26:ga   a23       ,   arn 1b1   ; PACK REAL: exp 10:= Radr;
    nkf  39        ,   dkf 3b1   ;    R:= N/factor; for exp 10 :=
a23:bt [exp 10]    t   -1        ;   exp 10 - 1 while exp 10 > 0 do
a27:mkf [q]        ,   hv  a23   ;    R:= R × (if pos exp then 10 else 0.1);
    grf  1b1                     ;    N:= R;
c49:qq   (e1)      t   -1        ; OUTPUT NUMBER: byte address:=
                                 ;    byte address - 1; OUTPUT LITERAL:
c67:hv   a28                LQA  ;    if introuble then goto AFTER OPERAND;
    bsn  (a2)      Xt  7         ;    if state > 7 then
    hv   a25                     ;    goto AFTER CONST OUT;
    arn  1b1       ,   pa  a18   ;    for i:= 0 step 10 until 30 do
a19:cl   10        X             ;       output (bits(i, i+9, n1));
    ck   -10       ,   hs  e3    ;
a18:btn            t   -150      ;
a17:pmn [kind]     DXV           ;
    hv   a19       X             ;
    hs   e3                      ;    output(kind)
a25:pm   (c1)      DXt 3         ; AFTER CONST OUT: operand:=
    ca   3                       ;    R:= operand + 3; if R = 3 then
a28:ps   c66       ,   hv  c6    ;    goto AFTER OPERAND;
    qq   (e1)      t   1         ;    byte address:= byte address + 1;
    hs   c7        ,   qq  s+d56 ;    error message({<const.});
```

```
c35:qq  (e1)     t   -1          ; INITIALIZE NUMBER:
                                 ;    byte address:= byte address - 1;
    grn  1b1     ,   pa  a15     ;    n1:= 0; nstate:= 0;
    pm   1b      ,   gm  3b1     ;    factor:= 1;
    hv   c37                     ;    goto NEXT OF NUMBER;
c38:arnf 3b1     ,   mkf 2b      ; DIGIT AFTER POINT:
    grf  3b1     ,   it  15      ;    factor:= factor x 10;
                                 ;    nstate:= 15; goto DIGIT 12;
c39:pa   a15     t   5           ; DIGIT BEFORE POINT: nstate:= 5;
    pm   1b1     ,   hs  c58     ; DIGIT 12: M:= n1 x 10 + (symbol - 57)
    gm   1b1                     ;    n1:= M
    pa   a15     t   35   NZ     ;    if overflow from n1 then
                                 ;       nstate:= 35;
c81:hv   c37                     ; BLIND: goto NEXT OF NUMBER;
c52:pa   a15     t   30          ; DIGIT 3: nstate:= 30;
    pm   2b1     ,   hs  c58     ;    n2:= n2 x 10 + (symbol - 57);
                                 ;    if overflow from n2 then
c53:pa   a15     t   35   NZ     ; ERROR 1: nstate:= 35;
    gm   2b1     ,   hv  c37     ;    goto NEXT OF NUMBER;
b k=e31, i=0                     ;
d i=e32                          ;    Alarm text used in:
d56: tconst.;                    ;    c48, 2a28
d e32=i                          ;
e                                ;

c40:pm   5b      ,   gm  1b1     ; TEN 1: n1:= 1;
c41:pa   a27     t   2b          ; TEN 2: q:= 10;
    grn  2b1     ,   it  20      ;    n2:= 0; nstate:= 20;
                                 ;    goto NEXT OF NUMBER;
c42:pan  a15     Vt  10          ; POINT: nstate:= 10;
                                 ;    goto NEXT OF NUMBER;
c43:pa   a27     t   7b          ; EXPMINUS: q:= 0.1
c44:pa   a15     t   25   NZ     ; EXPPLUS: nstate:= 25;
    hv   c37                     ;
c51:grn  1b1     ,   hs  e5      ; ERROR 2: n1:= 0;
                                 ;    error message(‡<const.‡);
c48:grn  2b1     ,   qq  s+d56   ; FINISH 2: n2:= 0;
c79:pmn  2b1     ,   cl  30      ; FINISH 3: Madr:= n2;
    hv   c51                NZ  ;    if n2 > 511 then goto ERROR 2;
    pa   a17     Xt  d11         ;    kind:= real; Radr:= Madr;
    hv   a26                     ;    goto PACK REAL;
c80:pa   a17     t   d12         ; FINISH 1: kind:= integer;
    hv   c49                     ;    goto OUTPUT NUMBER;
```

```
[3.8.66]
[GIER ALGOL 4, Pass 3, page 10, stack words 1]
[New state.9+switching part.19+stack part.29+output part.39]

      [stack: switching part, stack part, output]
      [; 4 and ;7]
      d d20=i-3                                  ;
[3]   qqf                                        ; trouble: -, -, -
[4]   qq                                 136.39,; thenst: -, -, end thenst
[5]   qq                                 137.39,; goto: -, -, end goto
[6]   qq                                  75.39,; assign: -, -, end assign
[7]   qq                                 171.39,; loop: -, -, end loop
[8]   qq                                 142.39,; single do: -, -, end single do
[9]   qq                                 141.39,; do: -, -, end do
[10]  qq                                 135.39,; elsest: -, -, end elsest
[11]  qq 27.9+ c4.19              +       21.39 ; begclean: OUT, -, ;
[12]  qq 27.9+ c4.19              +       21.39 ; beg block: OUT, -, ;
[13]  qq 27.9+ c4.19              +       21.39 ; beg body: OUT, -, ;
[14]  qq 27.9+ c4.19              +      170.39 ; of st: OUT, -, case semicolon
[15]  qq 28.9+c22.19              +        0.39 ; beg proc: PROC END, -,
[16]  qq 28.9+c23.19              +       29.39 ; core: ANOUT, -, core code end
[17]  qq                          +      155.39,; := switch: -, -, end switch

      [end 1 and end 2]
      d d21=i-3                                  ;
[3]   qqf                                        ; trouble: -, -, -
[4]   qq                                 136.39,; thenst: -, -, end thenst
[5]   qq                                 137.39,; goto: -, -, end goto
[6]   qq                                  75.39,; assign: -, -, end assign
[7]   qq                                 171.39,; loop: -, -, end loop
[8]   qq                                 142.39,; single do: -, -, end single do
[9]   qq                                 141.39,; do: -, -, end do
[10]  qq                                 135.39,; elsest: -, -, end elsest
[11]  qq 20.9+c23.19              +       15.39 ; beg clean: AN OUT, -, end clean
[12]  qq 20.9+c61.19              +       16.39 ; beg block: BLOCK COUNT, -, end block
[13]  qq 10.9+c24.19                           ; beg body: AN, -, -
[14]  qq 20.9+c23.19              +       26.39 ; of st: AN OUT, -, end case st

      [else 1 and else 2]
      d d22=i-2                                  ;
[2]   qq  3.9+c19.19+  1.29+             132.39 ; thenex: CH OUT, else ex, else ex
[3]   qq 34.9+c30.19                           ; trouble:  AN TROUBLE, -, -
[4]   qq  9.9+c19.19+ 10.29+              24.39 ; thenst: CH OUT, else st, else st
[5]   qq                                 137.39,; goto: -, -, end goto
[6]   qq                                  75.39,; assign: -, -, end assign
```

```
                [,7]
          d d23=i-17                    ;

[17]  qq  2.9+ c4.19+           169.39 ; := switch: OUT, -, case comma
[18]  qq  2.9+c19.19+ 38.29+    152.39 ; next colon: CHOUT, array comma, not first bound
[19]  qq  2.9+c19.19+ 38.29+    151.39 ; first colon: CHOUT, array comma, first bound
[20]  qq  2.9+c19.19+ 23.29+     35.39 ; [left: CHOUT, left, comma 1
[21]  qq  2.9+c19.19+ 24.29+     35.39 ; [subscr: CHOUT, subscr, comma 1
[22]  qq  2.9+c19.19+ 25.29+     35.39 ; [left or subs: CHOUT, left or sub, comma 1
[23]  qq  2.9+ c4.19+            36.39 ; , left: OUT, -, comma 2
[24]  qq  2.9+ c4.19       +     36.39 ; , subscr: OUT, -, comma 2
[25]  qq  2.9+ c4.19+            36.39 ; , left or subs: OUT, -, comma 2
[26]  qq  2.9+c34.19+ 27.29+     70.39 ; single comma: COUNTCHOUT, := for, simple for
[27]  qq  2.9+c34.19+ 26.29+     70.39 ; := for: COUNT CHOUT, single comma, simple for
[28]  qq  2.9+c34.19+ 27.29+     72.39 ; until: COUNT CHOUT, := for, stepelem
[29]  qq  2.9+c34.19+ 27.29+     73.39 ; while: COUNT CHOUT, := for, while elem
[30]  qq  2.9+ c4.19+           169.39 ; of exp: OUT, -, case comma

              [, 7 and parameter delimiter]

          d d41=i-31                    ;

[31]  qq  2.9+ c4.19+            34.39 ; (call: OUT, -, call param
[32]  qq  2.9+ c4.19+            34.39 ; (func: OUT, -, call param

         [right bracket 1]
         d d24=i-18                     ;
[18]  qq 13.9+c69.19+ 17.29+    152.39 ; next colon: BOUNDS, end bounds, not first bound
[19]  qq 13.9+c69.19+ 17.29+    151.39 ; first colon: bound base, first bound
[20]  qq 27.9+c25.19+  2.29+     32.39 ; [left: RIGHT, NEW OPERAND, ] one
[21]  qq  1.9+c25.19+  2.29+     32.39 ; [subscr: RIGHT, NEW OPERAND, ] one
[22]  qq  4.9+c25.19+  2.29+     32.39 ; [left or subs: RIGHT, NEW OPERAND, ] one
[23]  qq 27.9+c25.19+  2.29+     33.39 ; , left: RIGHT, NEW OPERAND, ] more
[24]  qq  1.9+c25.19+  2.29+     33.39 ; , subscr: RIGHT, NEW OP, ] more
[25]  qq  4.9+c25.19+  2.29+     33.39 ; , left or subscr: RIGHT, NEW OP, ] more

         [do 1]
         d d25=i-26                     ;
[26]  qq 27.9+c64.19+  8.29+    162.39 ; single comma: DO, single do, simple for do
[27]  qq 27.9+c64.19+  8.29+    162.39 ; := for: DO, single do, simpel for do
[28]  qq 27.9+c64.19+  8.29+    163.39 ; until: DO, single do, step elem do
[29]  qq 27.9+c64.19+  8.29+    164.39 ; while: DO, single do, while elem do

         [right parenthesis 2]
         d d26=i-30                     ;
[30]  qq  1.9+c25.19+  3.29+    168.39 ; of exp: RIGHT, NEW OPERAND, end case exp
[31]  qq 38.9+c68.19       +     31.39 ; (call: RIGHT CALL, NEW OPERAND, end call
[32]  qq  1.9+c25.19+  3.29+     31.39 ; (func: RIGHT, NEW OPERAND, end call
[33]  qq  1.9+c25.19+  3.29+    161.39 ; (subex: RIGHT, NEW OPERAND, )
```

```
       [then 1]
       d d27=i-34                                  ;
[34]   qq  5.9+c19.19+  2.29+  131.39 ; ifex: CH OUT, thenex, thenex
[35]   qq  8.9+c19.19+  4.29+   23.39 ; ifst: CH OUT, thenst, thenst

       [step 1]
       d d28=i-26                                  ;
[26]   qq  2.9+c19.19+ 36.29+  139.39 ; single comma: CHOUT, step, step
[27]   qq  2.9+c19.19+ 36.29+  139.39 ; := for: CH OUT, step, step

       [until 1]
       d d29=i-36                                  ;
[36]   qq  2.9+c19.19+ 28.29+  140.39 ; step: CH OUT , until, until

       [while 1]
       d d30=i-26                                  ;
[26]   qq  2.9+c36.19+ 29.29+   74.39 ; single comma: WHILE COUNT, while, while
[27]   qq  2.9+c19.19+ 29.29+   74.39 ; := for: CH OUT, while, while

       [:3]
       d d31=i-37                                  ;
[37]   qq  2.9+c19.19+ 19.29+   37.39 ; [arr: CH OUT, first colon, boundcolon
[38]   qq  2.9+c19.19+ 18.29+   37.39 ; array comma: CH OUT, next colon, bound colon

       [of]
       d d38=i-39                                  ;
[39]   qq 11.9+c23.19+         167.39 ; case exp: ANOUT, -, of exp
[40]   qq 26.9+c23.19+          25.39 ; case st: ANOUT, -, of st

       [for 3]
       d d39=i-8                                   ;
[8]    qq 20.9+c19.19+  7.29+  142.39 ; single do: CHOUT, loop, end single do
```

[trouble 1]

```
d d32=i-1                                          ;

[1]   qqf                                          ; else ex: -, -, -
[2]   qqf                                          ; then ex: -, -, -
[3]   qqf                                          ; trouble: -, -, -
[4]   qq 34.9+c18.19+  3.29+    41.39 ; then st: ENT OUT, trouble, trouble
[5]   qq 34.9+c19.19+  3.29+    41.39 ; goto: CH OUT, trouble, trouble
[6]   qq 34.9+c19.19+  3.29+    41.39 ; assign: CH OUT , trouble, trouble
[7]   qqf                                          ; loop:
[8]   qq 34.9+c18.19+  3.29+    41.39 ; single do: ENTOUT, trouble, trouble
[9]   qq 34.9+c18.19+  3.29+    41.39 ; do: ENT OUT, trouble, trouble
[10]  qq 34.9+c18.19+  3.29+    41.39 ; elsest: ENT OUT, trouble, trouble
[11]  qq 34.9+c18.19+  3.29+    41.39 ; beg clean: ENT OUT, trouble, trouble
[12]  qq 34.9+c18.19+  3.29+    41.39 ; beg block: ENT OUT, trouble, trouble
[13]  qq 34.9+c18.19+  3.29+    41.39 ; beg body: ENT OUT, trouble, trouble
[14]  qq 34.9+c18.19+  3.29+    41.39 ; of st: ENTOUT, trouble, trouble
[15]  qq 32.9+c28.19       +    41.39 ; beg proc: TRPROCEND, -, trouble
[16]  qq 32.9+c23.19       +    41.39 ; core: ANOUT, -, trouble
[17]  qqf                                          ; := switch: -, -, -
[18]  qq 32.9+c23.19+          41.39 ; next colon: ANOUT, -, trouble
[19]  qq 32.9+c23.19       +    41.39 ; first colon: AN OUT, -, trouble
[20]  qqf                                          ; [left: -, -, -
[21]  qqf                                          ; [subscr: -, -, -
[22]  qqf                                          ; [left or subs: -, -, -
[23]  qqf                                          ; , left:
[24]  qqf                                          ; , subscr:
[25]  qqf                                          ; , left or subscr:
[26]  qqf                                          ; single comma:
[27]  qqf                                          ; := for: -, -, -
[28]  qqf                                          ; until: -, -, -
[29]  qqf                                          ; while: -, -, -
[30]  qqf                                          ; of exp
[31]  qqf                                          ; (call: -, -, -
[32]  qqf                                          ; (func: -, -, -
[33]  qqf                                          ; (subex: -, -, -
[34]  qqf                                          ; if ex: -, -, -
[35]  qqf                                          ; if st: -, -, -
[36]  qqf                                          ; step: -, -, -
[37]  qq 32.9+c23.19       +    41.39 ; [arr: AN OUT, -, trouble
[38]  qq 32.9+c23.19       +    41.39 ; array comma: ANOUT, -, trouble
[39]  qqf                                          ; case exp:
[40]  qqf                                          ; case st
```

[3.8.66]
[GIER ALGOL 4, Pass 3, Page 14, control table for numbers]

[Input byte values from 56 to 68 have their control words c-marked (i.e. f and comma marked) and call for a special logic for analyzing numbers. While this logic is operating the input byte values are converted so as to give the appropriate entry of the following table. This is an action table having the current numberstate as the other argument. The action is given in the table as the machine address of the code relative to DIGIT 2 = c38. The number states are:

Number state	Position	
0	4	Before number
5	9	Following digit before point
10	14	Following point
15	19	Following digit after point
20	24	Following ten
25	29	Following exponent sign
30	34	Following digit after ten
35	39	In erroneous number

When an error has been detected the remaining part of the number is skipped. The error message is given on the following terminator.]

[Entry to table: direct converted]

```
d d2=i-56
d c39=c39-c38, c81=c81-c38, c52=c52-c38 ;
d c53=c53-c38, c40=c40-c38, c41=c41-c38 ;
d c42=c42-c38, c43=c43-c38, c44=c44-c38 ;
d c51=c51-c38, c48=c48-c38              ;
d c79=c79-c38, c80=c80-c38              ;
```

```
[56]  qqf c51.4+c80.9+c51.14+c48.19+c51.24+c51.29+c79.34+c51.39,;      <terminator>
[57]  qqf c39.4+c39.9+  0.14+  0.19+c52.24+c52.29+c52.34+c81.39,;  0   <digit>
[58]  qqf c42.4+c42.9+c53.14+c53.19+c53.24+c53.29+c53.34+c81.39,;  1     .
[59]  qqf c40.4+c41.9+c53.14+c41.19+c53.24+c53.29+c53.34+c81.39,;  2    10
[60]  qqf,                                                         ;  3
[61]  qqf,                                                         ;  4
[62]  qqf,                                                         ;  5
[63]  qqf c51.4+c80.9+c51.14+c48.19+c44.24+c51.29+c79.34+c51.39,;  6     +
[64]  qqf,                                                         ;  7
[65]  qqf,                                                         ;  8
[66]  qqf,                                                         ;  9
[67]  qqf c51.4+c80.9+c51.14+c48.19+c43.24+c51.29+c79.34+c51.39,;  .     -
[68]  qqf,                                                         ; 10
```

[3.8.66]
[GIER ALGOL 4, Pass 3, Page 15, main control table]

[Each input byte gives access to a word in the following table. If this word is NOT COMMA-MARKED the table contains NORMAL action words arranged as follows:

```
                Delimiter action word p
                    -       -     -   p-1
                . . . . . .
                    -       -     -   1
                Delimiter meaning word q
                    -       -     -   q-1
byte value:         -       -     -   1
                Delimiter meaning comment.
```

The delimiter meaning words in their bit no. w contain the number of the action word appropriate to the state w, in binary form, thus:

						Multiplier
q=1, p=1	Delimiter meaning word 1:					1
q=2, p=2 or 3	-	-	-	2:		1
	-	-	-	1:		2
q=3, p=4 to 7	-	-	-	3:		1
	-	-	-	2:		4
	-	-	-	1:		2

The action word number corresponding to the various states is also given in the delimiter meaning comment. Delimiter meaning word q is f-marked.

Delimiter action words come in two formats:

NO MARKS: SIMPLE ACTION WORDS

qq allowed operand.3+new state.9+switching part.19+stack part.29+output.39

When the program indicated in the switching part is entered we have

R: qq stack part.9+(output:512).10+new state.29+switching part.39

M: qq (output mod 512).9

qq(f) lim 1, qq lim 2 + base address.19

For a delimiter admitting the stack delimiters LOW to UP inclusive we have: lim 1 = -(512+LOW), lim 2 = lim 1 + UP.

Not f-marked words perform TEST FOR ELSE EXPRESSION. f-marked words perform TEST FOR PROCEDURE CALL. The base address points to the table of stack words.

If the word indicated by the input byte value is COMMA-MARKED we have a SPECIAL ACTION:

Not-f marked words supply:

qq switching part.19+parameter.29+output.39,

When the program indicated in the switching part is entered we have

R: qq parameter.9+(output:512).10+switching part.39

M: qq (output mod 512).9

f-marked words enter the number reading program INITIALIZE NUMBER, controlled by the control table for numbers.]

[3.8.66]
[GIER ALGOL 4, Pass 3, Page 16, main control table]

```
qq   7.3+ 6.9+ c4.19          +  144.39 ; +2: OUT, -, +
qq  15.3+ 6.9+c27.19+  504.29+  144.39 ; +1: PLUSMINUS, pos, +
qqf 31.5+ 1.7                          ; 1
qq   1.6                               ; 2: 72
[01111 12100 1 00000 00000 2 00000 00000 3 00000 00000]
qq   7.3+ 6.9+ c4.19          +  145.39 ; -2: OUT, -, -
qq  15.3+ 6.9+c27.19+  505.29+  145.39 ; -1: PLUSMINUS, neg, -
qqf 31.5+1.7                           ; 1
qq   1.6                               ; 2: 76
[01111 12100 1 00000 00000 2 00000 00000 3 00000 00000]
qq           c32.19           +   0.39, ; 77 CARRET: CR, -, CR
qq           c33.19           +   4.39, ; 78 short string: LITERAL , -, lit string
qq           c33.19+  512.29+   4.39, ; 79 long string: LITERAL, -, lit string
qq           c33.19           +   3.39, ; 80 boolean: LITERAL , -, lit string
qq  8.3+ 7.9+  c4.19          + 500.39 ; -,1: OUT, -, not
qqf 31.5+                              ; 1: 82
[01111 10000 1 00000 00000 2 00000 00000 3 00000 00000]
qq  8.3+27.9+ c77.19  +d33.29          ; goto 2: COMPL HEAD, unspec
qq  8.3+ 2.9+ c21.19+    5.29          ; goto 1: ENT, goto, -
qqf 12.11+3.28                         ; 1
qq  13.33                              ; 2: 86
[00000 00011 1 00000 00000 2 00000 00110 3 22020 00000]
qq  8.3+27.9+ c21.19+   14.29          ; begin 4: ENT, of st
qq  8.3+29.9+ c77.19+  d33.29          ; begin 3: COMPL HEAD, unspec
qq  8.3+28.9+ c21.19+   13.29          ; begin 2: ENT, beg body, -
qq  8.3+28.9+ c18.19+   11.29+   20.39 ; begin 1: ENT OUT, beg clean, begin
qqf 12.11+3.28+31.34                   ; 1
qq  1.26                               ; 4
qq  7.31+1.33                          ; 2: 93
[00000 00011 1 00000 00000 2 00000 04112 3 33131 00000]
qqf -520, qq d39.19-512                ; for 3: SEARCH STATEMENT
qq  8.3+27.9+ c77.19+  d33.29          ; for 2: COMPL HEAD, unspec
qq  8.3+22.9+ c31.19          +  138.39 ; for 1: FOR, -, for
qqf 12.11+3.28+1.20                    ; 1
qq  13.33+1.20                         ; 2: 98
[00000 00011 1 00000 00000 2 30000 00110 3 22020 00000]
qq  8.3+27.9+ c77.19+  d33.29          ; if 5: COMPL HEAD, unspec
qq  8.3+ 2.9+ c70.19+   34.29          ; if 4: CH, ifex, -
qq  8.3+ 2.9+ c18.19+   34.29+  129.39 ; if 3: ENT OUT, ifex, ifex
qq  8.3+ 2.9+ c70.19+   35.29          ; if 2: CH, ifst, -
qq  8.3+ 2.9+ c18.19+   35.29+  130.39 ; if 1: ENT OUT, ifst, ifst
qqf 5.4+27.31+1.33                     ; 1
qq  1.3+13.33                          ; 4
qq  5.4+1.9                            ; 2: 106
[00343 00002 1 00000 00000 2 00000 00110 3 55050 00000]
```

```
b11: qq c73.19-1.19-c76.19          ; modifier 2: value (c46)
qq  8.3+25.9+ c12.19+11.29          ; own 1: SET BLOCK, beg clean
qqf 17.32                           ; 1: 109
[00000 00000 1 00000 00000 2 00000 00010 3 00100 00000]
qq  8.3+ 6.9+  c4.19+       508.39 ; integer 4: OUT, -, opint
qq 8.3+15.9+c74.19+ d.29+d47.39-2.39; integer 3: FIRST SPEC, value yes, spec int-2
qq  8.3+17.9+ c13.19      +   9.39 ; integer 2: SET DECL, -, own integ
qq  8.3+19.9+ c12.19+11.29+   5.39 ; integer 1: SET BLOCK, beg clean, decl integ
qqf 1.28+15.33                      ; 1
qq  1.0-1.7                         ; 4
qq  1.25+13.33                      ; 2: 116
[04444 44400 1 00000 00000 2 00000 20010 3 33130 00000]
qq  8.3+ 6.9+  c4.19+       509.39 ; real 4: OUT, -, opreal
qq  8.3+15.9+ c74.19+ d.29+  51.39 ; real 3:FIRST SPEC, value yes, spec real-2
qq  8.3+17.9+ c13.19      +  10.39 ; real 2: SET DECL, -, own real
qq  8.3+19.9+ c12.19+11.29+   6.39 ; real 1: SET BLOCK, beg clean, decl real
qqf 1.28+15.33                      ; 1
qq  1.0-1.7                         ; 4
qq  1.25+13.33                      ; 2: 123
[04444 44400 1 00000 00000 2 00000 20010 3 33130 00000]
qq  8.3+ 6.9+  c4.19+       510.39 ; Boolean 4: OUT, -, op boolean
qq  8.3+15.9+ c74.19+ d.29+  52.39 ; Boolean 3: SET DECL, value yes, spec bool - 2
qq  8.3+17.9+ c13.19+        11.39 ; Boolean 2: SET DECL, -, own bool
qq  8.3+19.9+ c12.19+11.29+   7.39 ; Boolean 1: SET BLOCK, beg clean, decl bool
qqf 1.28+15.33                      ; 1
qq 1.0-1.7                          ; 4
qq  1.25+13.33                      ; 2: 130
[04444 44400 1 00000 00000 2 00000 20010 3 33130 00000]
qq  8.3+14.9+c75.19+ 0.29+  12.39  ; procedure 4: SEC SPEC, val no, proc-simple spec
qq  8.3+14.9+c74.19+ 0.29+  61.39  ; procedure 3: FIRST SPEC, value no, spec proc-2
qq  8.3+16.9+c14.19       +  38.39  ; procedure 2: ADD DECL PROC, -, simple-proc
qq  8.3+16.9+c11.19+11.29+  42.39  ; procedure 1: SET BLOCK PROC, beg clean, proc
qqf 1.28+15.33                      ; 1
qq  1.15                            ; 4
qq  1.19+13.33                      ; 2: 137
[00000 00000 1 00000 40002 2 00000 00010 3 33130 00000]
qq  8.3+14.9+c75.19+ 0.29+   8.39  ; array 4: SEC SPEC, value no, array-simple spec
qq  8.3+14.9+c74.19+ 0.29+  59.39  ; array 3: FIRST SPEC, value no, spec array-2
qq  8.3+24.9+c15.19       +   7.39  ; array 2: ADD DECL, -, simple-array decl
qq  8.3+24.9+c12.19+11.29+  13.39  ; array 1: SET BLOCK, beg clean, real array
qqf 1.28+15.33                      ; 1
qq  1.15                            ; 4
qq  1.19+13.33                      ; 2: 144
[00000 00000 1 00000 40002 2 00000 00010 3 33130 00000]
b13: qq 1.19-50.35+7.39             ; modifier 1: specification
                                    ; spec-2 is placed in pos.19 (1c74, 1c75)
```

[3.8.66]
[GIER ALGOL 4, Pass 3, Page 18, main control table]

```
qq  8.3+14.9+c74.19+ 0.29+   65.39  ; switch 2: FIRST SPEC, value no, spec switch-2
qq  8.3+23.9+c83.19+11.29+  d19.39  ; switch 1: SWITCH, beg clean, begin proc
qqf 1.28+1.32                       ; 1
qq  13.33                           ; 2: 149
[00000 00000 1 00000 00000 2 00000 00010 3 22120 00000]
qq  8.3+ 6.9+ c4.19+        511.39  ; string 2: OUT, -, opstring
qq  8.3+14.9+c74.19+ 0.29+   53.39  ; string 1: FIRST SPEC, value no, string-2
qqf 13.33                           ;
qq  1.1-1.7                         ; 2: 153
[02222 22200 1 00000 00000 2 00000 00000 3 11010 00000]
qq  8.3+14.9+c74.19+ 0.29+   54.39  ; label 1: FIRST SPEC, value no, spec label-2
qqf 13.33                           ; 1: 155
[00000 00000 1 00000 00000 2 00000 00000 3 11010 00000]
qq  8.3+12.9+c46.19+ 0.29+   66.39  ; value 1: VALUE, value no, undeclspec - 2
qqf 9.33                            ; 1: 157
[00000 00000 1 00000 00000 2 00000 00000 3 10010 00000]
qq 12.3+27.9+c78.19+d33.29          ; ; 7: SEMICOLON, unspec
qq  4.3+31.9+c76.19                 ; ;6: SPEC COMMA, -, -
qq  8.3+28.9+ c1.19                 ; ;5: NEXT, -, -
qq  -515, qq d20.19-498             ; ;4: SEARCH IN EXPRESSION
qq  8.3+30.9+ c1.19                 ; ;3: NEXT, -, -
qq  4.3+31.9+c16.19                 ; ;2: FORMALLIST, -,
qq  4.3+28.9+c20.19                 ; ;1: DECL, -, -
qqf 3.9+9.13+15.20+7.29-1.35+1.38   ; 1
qq  15.4+31.10+15.15+1.20+3.28+7.32+1.34+1.38 ; 4
qq  7.10+5.14+13.18+1.20+3.28+3.31+7.35+1.38  ; 2: 167
[04444 04477 1 70656 62131 2 70000 00770 3 77537 30070]
qq   8.3+28.9+c77.19+d44.29         ; end 3: COMPL HEAD, spec gen
qq  -515, qq d21.19-501             ; end 2: SEARCH IN EXPRESSION
qqf -515, qq d21.19-501             ; end 1: SEARCH STATEMENT
qqf 3.9+1.20+3.28+7.34+1.38         ; 1
qq  15.4+3.7+1.33                   ; 2: 172
[02222 02211 1 00000 00000 2 10000 00110 3 00131 00010]
qq  -514, qq d22.19-508             ; else 2: SEARCH IN EXPRESSION
qqf -514, qq d22.19-508             ; else 1: SEARCH STATEMENT
qqf 1.8+1.20+1.38                   ; 1
qq  15.4+7.7                        ; 2: 176
[02222 22210 1 00000 00000 2 10000 00000 3 00000 00010]
```

```
qq  4.3+27.9+c77.19+d33.29                ; (5: COMPL HEAD, unspec
qq  8.3+ 2.9+c21.19+ 30.29                ; (4: ENT, of exp
qq 12.3+ 2.9+c26.19+ 33.29+  160.39 ; (3: LEFT PARENT, subex, (
qq  4.3+ 2.9+c18.19+ 31.29+   38.39 ; (2: ENT OUT, (call, beg call
qq  4.3+21.9+c16.19                       ; (1: FORMAL LIST, -, -
qqf 31.5+3.7+1.16+3.31                    ; 1
qq  1.11+3.31                             ; 4
qq  1.0-1.9+3.28                          ; 2: 184
[03333 33322 1 04000 01000 2 00000 00220 3 55000 00000]
b25: qqf -515, qq d20.19-498              ; semicolon 7: SEARCH STATEMENT
qq  4.3+27.9+c77.19+d33.29                ; :6: COMPL HEAD, unspec
qq  4.3+36.9+ c1.19                       ; :5: NEXT, -, -
qq  4.3+ 9.9+ c4.19       +     8.39 ; :4: OUT, -, decl label
qq  -549, qq d31.19-511                   ; :3: SEARCH IN EXPRESSION
qq  4.3+ 8.9+ c4.19       +     8.39 ; :2: OUT, -, decl label
qq  4.3+27.9+ c4.19       +     8.39 ; :1: OUT, -, decl label
qqf 7.3+1.6+3.28+1.36                     ; 1
qq     + 1.9+3.31+1.36                    ; 4
qq  7.3+5.8+3.31                          ; 2: 194
[03330 03024 1 00000 00000 2 00000 00110 3 66000 05000]
qq  -538, qq d28.19-511                   ; step 1: SEARCH IN EXPRESSION
qqf 7.3+1.6                               ; 1: 196
[01110 01000 1 00000 00000 2 00000 00000 3 00000 00000]
qq  -548, qq d29.19-512                   ; until 1: SEARCH IN EXPRESSION
qqf 7.3+1.6                               ; 1: 198
[01110 01000 1 00000 00000 2 00000 00000 3 00000 00000]
qq  -538, qq d30.19-511                   ; while: SEARCH IN EXPRESSION
qqf 7.3+1.6                               ; 1: 200
[01110 01000 1 00000 00000 2 00000 00000 3 00000 00000]
qq  -530, qq d24.19-505                   ; ] 1: SEARCH IN EXPRESSION
qqf 7.3+1.6                               ; 1: 202
[101110 01000 1 00000 00000 2 00000 00000 3 00000 00000]
qq  4.3+27.9+c77.19+d33.29                ; [5: COMPL HEAD, unspec
qq  4.3+ 2.9+c18.19+ 21.29+   40.39 ; [4: ENT OUT, [subscr, [
qq  4.3+ 2.9+c18.19+ 22.29+   40.39 ; [3: ENT OUT, [left or subs, [
qq  4.3+ 2.9+c18.19+ 20.29+   40.39 ; [2: ENT OUT, [left, [
qq  4.3+ 2.9+c17.19+ 37.29                ; [1: DECL ENT, [arr, -
qqf 1.4+1.24+3.31                         ; 1
qq  7.3+7.7+3.31                          ; 4
qq  1.4+3.9+3.28                          ; 2: 210
[04443 44422 1 00000 00000 2 00001 00220 3 55000 00000]
```

```
qq  -529, qq d23.19-497                ; ,7: SEARCH IN EXPRESSION
qq  8.3+24.9+ c1.19                    ; ,6: NEXT, -, -
qq  4.3+14.9+c76.19                    ; ,5: SPEC COMMA, -, -,
qq  4.3+24.9+ c1.19                    ; ,4: NEXT, -, -
qq  4.3+21.9+c72.19                    ; ,3: FORMAL, -, -
qq  4.3+12.9+c76.19                    ; ,2: SPEC COMMA, -, -
qq  4.3+17.9+ c1.19                    ; ,1: NEXT, -, -
qqf 7.3+3.7+3.15+21.21                 ; 1
qq  7.3+3.7+7.15+1.24                  ; 4
qq  7.3+3.7+3.13+1.21                  ; 2: 220
[07770 07700 1 00265 50101 2 03004 00000 3 00000 00000]
qq  4.3+27.9+c77.19+d33.29             ; := 5: COMPL HEAD, unspec
qq  4.3+ 2.9+c18.19+ 27.29+   71.39 ; := 4: ENT OUT, := for, := for
qq  6.3+ 4.9+ c4.19       +   76.39 ; := 3: OUT, -, :=
qq  6.3+ 4.9+c18.19+  6.29+   77.39 ; := 2: ENT OUT , assign, first:=
qq  4.3+ 2.9+c84.19       +  153.39 ; := 1: SWITCH ASSIGN, -, of switch
qqf 1.4+1.23+3.31                      ; 1
qq  1.22+3.31                          ; 4
qq  1.4+3.9+3.28                       ; 2: 228
[00003 00022 1 00000 00000 2 00410 00220 3 55000 00000]
qq                                     ; not used
qq  -546, qq d27.19-511                ; then 1: SEARCH IN EXPRESSION
qqf 7.3+3.7                            ; 1: 231
[01110 01100 1 00000 00000 2 00000 00000 3 00000 00000]
qq  8.3+32.9+c22.19       +    0.39 ; trouble 7: PROC END, - , end proc
qq  8.3+33.9+c16.19                    ; trouble 6: FORMAL LIST, -, -
qq  8.3+32.9+c23.19       +   41.39 ; trouble 5: AN OUT, -, trouble
qq  8.3+32.9+c20.19                    ; trouble 4: DECL, -, -
qq  8.3+33.9+ c1.19                    ; trouble 3: NEXT, -, -
qq  8.3+32.9+ c4.19       +   41.39 ; trouble 2: OUT, -, trouble
qqf -513, qq d32.19-473                ; trouble 1: SEARCH STATEMENT
qqf31.5+31.10+27.15+1.18+15.23+59.31+7.38; 1
qq  1.10+3.17+1.19+3.24                    ; 4
qq  1.10+31.16+9.21+1.25+3.31          ; 2: 241
[01111 11111 1 71323 36434 2 13154 21110 3 33000 01110]
qq  -538, qq d25.19-509                ; do 1: SEARCH IN EXPRESSION
qqf 7.3+3.7                            ; 1: 243
[01110 01100 1 00000 00000 2 00000 00000 3 00000 00000]
```

```
qq  8.3+ 6.9+ c4.19+          506.39  ; abs 1: OUT, -, abs
qqf 1.0-1.7                           ; 1:245
[01111 11100 1 00000 00000 2 00000 00000 3 00000 00000]

qq  8.3+27.9+c77.19+d44.29            ; code 2: COMPL HEAD, specgen
qq  8.3+36.9+ c4.19       +  154.39  ; code 1: OUT, -, code
qqf 3.9+3.28+1.37                     ; 1
qq  13.33                             ; 2: 249
[00000 00011 1 00000 00000 2 00000 00110 3 22020 00100]

qq  8.3+37.9+c18.19+ 16.29+   28.39 ; core 1: ENT OUT, core, core code
qqf 1.28                              ; 1: 251
[00000 00000 1 00000 00000 2 00000 00010 3 00000 00000]

qq  8.3+27.9+c77.19+d33.29            ; case 3: COMPL HEAD, unspec
qq  8.3+ 2.9+c18.19+ 40.29+  165.39 ; case 2: ENT OUT, case st, case st
qq  8.3+ 2.9+c18.19+ 39.29+  d48.39 ; case 1: ENT OUT, case exp, case exp
qqf 14.5+13.33                        ; 1
qq  1.9+3.28+13.33                    ; 2: 256
[00111 00002 1 00000 00000 2 00000 00220 3 33030 00000]

qq  -551, qq d38.19-511               ; of 1: SEARCH IN EXPRESSION
qqf 7.3+3.7                           ; 1: 258
[01110 01100 1 00000 00000 2 00000 00000 3 00000 00000]

qq  8.3+ 6.9+ c4.19+          507.39 ; round 1: OUT, -, round
qqf 1.0-1.7                           ; 1:260
[01111 11100 1 00000 00000 2 00000 00000 3 00000 00000]

qq  8.3+ 6.9+ c4.19+          501.39 ; entier 1: OUT, -, entier
qqf 1.0-1.7                           ; 1: 262
[01111 11100 1 00000 00000 2 00000 00000 3 00000 00000]

qq  4.3+38.9+c82.19+999.29+   27.39 ; beg code 1: CODE, code begin, code end
qqf 7.34+1.36                         ; 1: 264
[00000 00000 1 00000 00000 2 00000 00000 3 00111 01000]
```

[3.8.66]
[GIER ALGOL 4, Pass 3, Page 22, main control table]

```
qq  c60.19+ 3.29+    3.39, ; 265: true: LOGIC VALUE, true, lit bool
qq  c60.19+ 0.29+    3.39, ; 266: false: LOGIC VALUE, false, lit bool
qq  c29.19+ 6.29+  146.39, ; 267: × : BINARY, -, ×
qq  c29.19+ 6.29+  147.39, ; 268: / : BINARY, -, /
qq  c29.19+ 6.29+  149.39, ; 269: ↑ : BINARY, -, ↑
qq  c29.19+ 6.29+  148.39, ; 270: ↓ : BINARY, -, ↓
qq  c29.19+ 1.29+  492.39, ; 271: < : BINARY, -, <
qq  c29.19+ 1.29+  493.39, ; 272: ≤ : BINARY, -, ≤
qq  c29.19+ 1.29+  494.39, ; 273: = : BINARY, -, =
qq  c29.19+ 1.29+  495.39, ; 274: ≥ : BINARY, -, ≥
qq  c29.19+ 1.29+  496.39, ; 275: > : BINARY, -, >
qq  c29.19+ 1.29+  497.39, ; 276: ≠ : BINARY, -, ≠
qq  c29.19+ 1.29+  156.39, ; 277: ∧ : BINARY, -, ∧
qq  c29.19+ 1.29+  157.39, ; 278: ∨ : BINARY, -, ∨
qq  c29.19+ 1.29+  159.39, ; 279: ≡ : BINARY, -, ≡
qq  c29.19+ 1.29+  158.39, ; 280: => : BINARY, -, =>
qq  c29.19+ 6.29+  143.39, ; 281: mod: BINARY, -, mod
qq  c29.19+ 1.29+  150.39, ; 282: shift: BINARY, -, shift
qq  c62.19,                ; endpass: ENDPASS, -, -
qq  4.3+21.9+c72.19        ; param delim 2: FORMAL, -, -
qq  -543, qq d41.19-511    ; param delim 1: SEARCH IN EXPRESSION
qqf 7.3+3.7                ; 1
qq  1.21                   ; 2: 287
[01110 01100 1 00000 00000 2 02000 00000 3 00000 00000]
qq  -542, qq d26.19-509    ; )2: SEARCH IN EXPRESSION
qq  1.1+18.9+c72.19        ; )1: FORMAL, -, -
qqf 1.21                   ; 1
qq  7.3+3.7                ; 2: 291
[02220 02200 1 00000 00000 2 01000 00000 3 00000 00000 ]

d1: qq [pass sum]          ;

d e22=k-e14,e47=j          ; set load parameters
b k=e23,i=0                ; load segment word 5
i=5e21                     ;
qq e16.9+1d1.19-e16.19+1.20+c63.39f ;
e                                   ;

e                          ; final end pass 3
s
```

[13.6.66]

[GIER Algol 4, pass 4, page 1]

```
b k=e22+e14, i=e16-e47, a22, b15, c48, d20 ; drum block head pass 4
i=e16                                      ;
```

[Use of indicator:

Name:	true=	init to:	comment:
in proc :	NTA	f	t whenever inside a proc declaration
in head :	LTB	f	Set to f on end proc or after begin. Tested on any declaration, bounds, end stack code, begin block, or, specs and if f then butput (end head); in head:= t;
active :	LZA	f	Used by the bypass logic which generates jumps around procedure declarations and sets the return point from local declaration
in block :	LOB	f	The word end bits is used as a stack of bits which keeps track of the current interpretation of begin. On ends endbits is shifted right 1 and bit 0 set to 1 if endblock or endproc, to 0 if end clean, and in block is set accordingly. A begin is a begin clear if -, in block. After begins are processed endbits is shifted 1 left and inblock:= bit (0) = 1;
warning :	LPB	f	Used by the logic which generates while label and prepass.
in trouble:	LQB	f	Used to prevent output of literals when in trouble.

marks of current actionword are set in RC by the central logic]

```
[Predefinitions, mostly output values]
d3 = 194 ; goto bypass
d4 = 195 ; bypass label
d5 = e20 ; initial top of use stack, not output.
d8 = 16  ; end decl
d9 = 511 ; search, stack value
d11= 64  ; take array
d12= 80  ; formal array
d13= 15  ; end head
d14= 16  ; end block, input value
d19= 2   ; end pass
d20= 24  ; end bound head
```

[The following a, b, and c names do not appear in natural order
c43 (after c9), c44 (after c17), c45 (after c31), a17 (after c22),
b15 (after c14), c47(after c13), c48(after c19), a2, a22(after a)]

```
b:   qq       [endbits]     ; c21, 1c21, c26, 1c26
[1]  qq       [owns]        ; table
[2]  qq       [variables]   ; 2c20,16c20,c25,3c25,3c31,5c31,6c31,table
[3]  qq       [locals]      ; c18,1c20,2c20,15c20,1c25,6c36,table
[4]  qq       [counter]     ; 7c22,1c23,2c23,c27,2c28,c29,1c29,c30,2c31

c1:
b1:  pmn [par 3]DX          ; stack out next:
     hs  c7                 ;   stack(par 3);
c2:
b2:  pmn [par 1]DX          ; out next:
     hs  e3           NZ    ;   Raddr:= par 1; if R ≠ 0 then output (R);
c3:  pmn(e1)  X  1          ; next:
     hs  e2           LA    ;   Raddr:= next byte;
a:   ga  b3   V      NT     ; after CR: if Raddr > maxinterest then
     ps  c3-1 , hv  e3      ;     begin set return (next); goto output end;
b3:  bs  d14  V d1   NZ     ;   if R = 0 then
     ps  a-1  , hv  a19     ;     begin set return(after CR); goto count CR end;
     ps  c3-1 , hv  e3      ; start: R:= set RC (table[Raddr]);
a21: pmn(b3)  X d2   IRC    ; rep: par 1:= part 1 (R);
a22: ga  b2                 ;   if LA then
     gt  b4   V      LA     ;     begin
     gt  r    , hvn -1      ;     par 3:= part 2 (R); par 4:= part 4 (R)
     ck  20   , ga  b1      ;     end;
b4:  gt  b13  , hvn -1      ;   M:= R:= 0; goto part 2 (R);

c4:  pmn d3   DXV    IZA    ; set active: output (goto bypass);
                            ;   active:= t; return;
c5:  arn d4   D      IZA    ; set not active: output (bypass label);
     hv  e3                 ;   active:= f; return;


c6:  arn(b1)  D             ; output par 3· output (par 3);
     hv  e3                 ;   return;


c7:  gr  p1           M     ; stack: stack top:= stack top + 1;
     pp  p1   , it (b5)     ;   core[stack top] M:= R;
                            ;   if stack top + 2 > use top then
c8h: bs  p2   , hs  e5      ; stack alarm: mess (‹<stack›, 2, 0);
     hr  s1   , qqn e34     ;   return;


c9:  arn p    , pp  p-1     ; byte unstack: R:= store [stack top];
     hr  s1                 ;   stack top:= stack top - 1 return;


c43: arn -d13 D      ITB    ; set in head: in head:= t;
     mt  c43  , hv  e3      ;   Raddr:= end head; goto output;
```

```
[Search use stack: search in use stack from use top to first block stop
 for identifier given in Raddr. Goes back with:
 if not found: R unchanged, entry = address of first blockstop, s = 0,
 if     found: R = 0, entry = addr where found, s = number of subscr]

c10: gs  b6    , ps  a1    ; search use stack: save s; s:= subcr table;
b5:  hv  d5[usetop]        ;   goto core [use top]; comment return to here
b6:  hv  -1    t   1       ;     with entry, R and s set; goback;
                           ; subscr table: comment
a1:  gs  b7    , hr  b6    ; 0   A search in the use stack terminates
     gs  b7    , hr  b6    ; 1   always in the entry in this table
     gs  b7    , hr  b6    ; 2   coresponding to the number of indices
     gs  b7    , hr  b6    ; 3   found (or 0).
     gs  b7    , hr  b6    ; 4   The following 3 formats may be encountered
     gs  b7    , hr  b6    ; 5   in the use stack during search:
     gs  b7    , hr  b6    ; 6   blockstop:  qq       , hs  s
     gs  b7    , hr  b6    ; 7   cancelled entry  qq
     gs  b7    , hr  b6    ; 8   normal entry: ca <ident>, hsn s<no of subscr>

a2:  ca        , hsn s -1  ; entry word
a3:  qq        , hs  s     ; use block stop


c11: ps  c2-1              ; literal: set return (out next);

c12: hs  c13               ; copy lit: copy 1;
     hs  c13               ;   copy 1;
     hs  c13               ;   copy 1;

c13: arn(e1)   t   1       ; copy 1:
     hs  e2          LA    ;   Raddr:= next byte;
     hv  e3          NQB   ;   if -, in trouble then goto output;
     hr  s1                ;   return;
```

[13.6.66]

[GIER Algol 4, pass 4, page 4]

```
b  a5                          ; block for finis pass 4

c47: grn e20            MC     ; finis pass 4: stack [top core] MC:= 0; inf 1:= 0;
                               ;   M:= 0;
a1:  pm  d15   t   1           ;   for i:= base stack,
     psn(a1)   VX  d18 LC      ;     i+1 while marks [i] ≠ Cmark do
     hv  a1                    ;       inf 1:= inf 1 + 1;

     gs  2e4   , hs  c16       ;   Raddr:= next byte;
                               ;     comment firstident - 1;

a2:  gm  1e20  t  -1    MB     ;   for i:= topcore step -1 until first ident-1 do
     nc (a2)   , hv  a2        ;     pass 5 table [i] MB:= 0;
     hs  e3             IQB    ;   output (Raddr); trouble:= true;

a3:  hs  c13                   ; set std proc:
     hv  a5    X         NT    ;   for Raddr:= next byte while Raddr > 511 do
     ga  a4    , ck  10        ;     begin
a4:  gm[ident] X         MC    ;     ident:= Raddr;
     is (a4)   , it  s512      ;     pass 5 table[ident] MC:=
     pa (a4)   , hs  c13       ;       M + inpos 29 (next byte) +
     ck  20    , ac (a4)       ;       in pos 9 (ident - 512);
                               ;     M:= in pos 39 (ident);
     hv  a3                    ;     end;
a5:  ck  -10   , hs  e3        ;   output (M shift 10);
     srn(1b)   D   2           ;   owns:= owns + 2;
     sr  3e4   , hs  e3        ;   output(-owns - max block level);
     arn 1b    , hs  e3        ;   output(owns);
     hhn e29                   ;   R:= 0; goto new segm;


e
```

```
c14: pa  4b              IQB ; trouble: counter:= 0; introuble:= t;
                             ;
a4:  hs  c16                 ; loop trouble: Raddr:= next not CR;
     hv  a4                LT ;   if Raddr > max out of trouble then
     ga  b15   , it  d6      ;     goto loop trouble;
b15: bs  -1    , hv  a4      ;   if Raddr ≤ max literal then
     it (b15)  , bs  d7      ;     begin set return (loop trouble)
     ps  a4-1  , hv  c12     ;       goto copy lit end;
     qq (e1)   t   -1  IQB ;   reset input; in trouble := f;
     hv  c3                  ;   goto next;



c16: arn(e1)   t   1         ; next not CR:
     hs  e2               LA ;   Raddr:= next byte;
     hr  s1               NZ ;   if R ≠ 0 then return;
a19: arn a5    , sc  e4      ; count CR:
     hsn e3                  ;   output (0);
     hv  c16                 ;   goto next not CR

a5: m                    1   ;

c17: pm  d8    DV        LOB ; begin: if -, inblock then goto decrease ends
     pa  b2    , hv  c21     ;   begin par 1:= 0; goto decrease ends end;
     hs  c43                NTB ; unstack block: if -, in head then set in head;
     hs  e3    X             ;   output(end decl);
a6:  arn p     , pp  p-1     ; loop unstack: Raddr:= unstack byte;
a7:  hh  a10                LZ ; test block byte: if R = 0 then goto block stop;
     nc  d9                 NT ;   if Raddr ≠ search then
     ps  a6-1  , hv  e3      ;     begin set ret(loop unstack); goto output end;


a8:  arn p     , pp  p-1     ; search use: Raddr:= unstack byte
     hv  a7                 NT ;   if Raddr < 512 then goto test block byte;
     hs  c10                   ;   search use stack;
     hh  a9                 NZ ;   if found then
b7:  gr [entry]              M ;     begin cancel(entry); Raddr:= stack byte end;
a9h: arn p1    , ps  a8-1    ;   set return (search use); goto output;

a10h:hv  e3    , hs  c9      ; block stop:
     ga  b14   , hs  c9      ;   last decl:= byte unstack;
     ga  b10   , hs  c9      ;   locals 1:= byte unstack;
     pi  0.3   V 959  NRB ;   variables 1:= byte unstack;
     ga  b11   , hv  c1      ;   if LRB then goto stack out next;
     ga  b11   , hv  c20     ;     comment : LRB = end specs;
                             ;   inhead:= f; goto out of block
```

[13.6.66]

[GIER Algol 4, pass 4, page 6]

```
c44: pp  p-1                ; no par type pr: stack top:= stack top - 1;
c18: hs  c13                ; par type pr: copy 1:
     qq (e1)   t   -1       ;   reset input; locals := locals + 1;
     qq (3b)   V   1        ;   goto out of block;

c19: pp  p-1                ; no par pr: stack top:= stack top - 1;
c48: arn(b12)   D -1   ITA  ; par pr: proclevel:= proclevel - 1;
                            ;   in proc:= proclevel ≥ 0;
c20: hs  c5            LZA  ; out of block: if active then set not active
     srn 3b    , sr  2b     ;   output (-locals - variables);
     hs  e3                 ;   output (-locals);
     srn 3b    , hs  e3     ;   comment base work and variable address
b8:  can 0[block level]t-1   ;   block level:= block level - 1;
     hv  c47                ;   if blocklevel = 0 then goto finis pass 4;
     hv  a12           LTA  ;   if in proc then collaps usage stack
                            ;     begin
     hsn c10                ;     R:= 0; search use stack; found:= t;
     grn(b5)   t   -1  MB   ;     stack inuse (special stop);
     ps (b7)   , gs  b9     ;     i:= entry; comment last block stop;
     ps  s1    , gs  b5     ;     top use:= i + 1
                            ;   loop collaps: if R ≠ 0 then search use stack;
a11: hs  c10           NZ   ;     if -, found then stack en use (R);
     ud  a14           NZ   ;     i:= i - 1;
b9:  arn -1    t   -1       ;     R:= use stack[i];
     hv  a11           NB   ;     if R ≠ special stop then goto loop collaps;
a12:                        ;     end;
b10: it [locals 1] , pa 3b  ;   locals:= locals 1;
b11: it [variables1],pa 2b  ;   variables:= variables 1;

c21: arn b     , cl  1      ; decrease ends: unstack end bit;
     gr  b             IOB  ;   block:= first end bit = 1;
     hv  c32           LRA  ;   if LRA then goto declare proc;
     hvn c2            IZA  ;   active:= t; goto outnext;
                            ;   comment NRA = begin;

c22: hs  c16                ; beg subscr: Raddr:= next not CR;
a17: qq (e1)t-1[excbya16]   ;   reset input;
     hv  c23           LTA  ;   if -, in proc then goto counter out;
     hs  c10                ;   search use stack;
     hv  c23           LZ   ;   if found then goto counter out;
     is (4b)   , it  s1     ;   part 2 of use entry word:= counter + 1;
     pt  a2    , ar  a2     ;   R:= R + use entry word;
   [a14 exec. by 1a11,7c25];   use top:= use top - 1;
a14: gr (b5)   t   -1  MA   ;   store[use top] MA:= R;
     it (b5)   , bs  p2     ;   if usetop < stack top + 2 then
     hh  c8                 ;   goto stack alarm;


c23: hs  c9    , qq  c2-1   ; counter out: M:= byte unstack;
     pm (4b)   DX   1       ;   R:= counter + 1; set return(out next);
     gm  4b    , hv  e3     ;   counter:= M; goto output
```

```
c24: hs  c43          NTB ; end proc: if -, in head then set in head
     hs  c5           LZA ;   if active then set not active;
     hs  c13              ;   copy 1; in head:= f;
b12: arn -1[proclev]D 1ITC ;   proc level:= proc level + 1; inproc:= t

c25: arn 2b    , hs  c7    ; end block:
     arn 3b    , hs  c7    ;   stack(variables); stack(locals);
     arn(b14)  D           ;   stack (last decl);
     pa  2b    , hs  c7    ;   variables:= locals:= last decl:= 0;
     psn(b8)   t   1   IZA ;   block level:= block level + 1;
     arn a3    , it (3e4)  ;   if block level > max block level then
     bs (b8)   , gs  3e4   ;     max block level:= block level;
     ud  a14         NTA   ;   active:= t;
     pa  3b    , hsn c7    ;   if inproc then stackuse (useblock stop);
     pa  b14   , arn a5    ;   stack(0); R39:= 1;
                           ;
c26: ar  b     , ck  -1    ; end clean: stack end bit (R39);
     gr  b           IOB   ;   block:= first end bit = 1;
     ck  -1                ;   if endbit 40 = 1 ∨ block level > 32 then
     bs (b8)   t   32  NO  ;     mess({<begin ends}, 2, 0);
     hs  e5                ;   goto outnext;
     hv  c2    , qqn d10   ;

c27: arn 4b    , pa  4b    ; start count:
     ps  c2-1  , hv  c7    ;   stack(counter); counter:= 0; goto outnext;


c28: hs  c43          NTB ; bounds: if -, inhead then set inhead;
     hs  c4           NZA ;   if -, active then set active;
     pa  4b    , hv  c2    ;   counter:= 0; goto outnext;


c29: bs (4b)   t   7      ; count index:
     pa  4b    , hs  e5   ;   if counter > 7 then
     hv  c30   , qq  sd16 ;     begin counter:= 0; mess({<index}, 0, 1) end;

c30: qq (4b)   t   1      ; count param: counter:= counter + 1;
     hv  c2               ;   goto out next;
```

```
c31: pa  b14   , can(4b)   ; declare array: last decl:= 0;
     arn a15   , hv  a22   ;   if counter = 0 then
     hs  c35   , qq  d20   ;     begin R:= is undeclared; goto rep end;
     gs  b1    , hs  c6    ;   test decl; par 3:= endboundhead;
     arn 4b    , hs  c7    ;   output par 3; stack counter;
     ac (2b)   D   1       ;   variables:= variables + counts + 1
     ps  c2-1  , hv  c34   ;   set return(outnext); goto stack and copy

c45: hs  c35   , qq  i+2   ; declare label: test decl;
     gs  a16   , hv  c34   ;   set return to here after one stack and copy in a16;
     pa  a16   t   c34     ; here: reset a16;
     hv  c2                ;   goto outnext;

c32: pa  b5    t d5    LTA ; declare proc:
     arn(b2)   D           ;   if -, in proc then usetop:=  initial usetop;
     pa  b14   , hs  e3    ;   last decl:= 0; output (par 1);

c33: hs  c35   , qq  c3-1  ; declare: test decl; set return (next);
     hs  c43          NTB ;   if -, in head then set in head;


c34: hs  c16               ; stack and copy: Raddr:= next not CR;
     hh  a16          NT   ;   if R < 0 then
     hs  c7                ;     begin
     hs  e3                ;       stack (Raddr); output (Raddr);
b13h:it  1     , qq ([par 4]);     counts [par 4]:= counts [par 4] + 1;
                           ;     goto stack and copy;
a16: hv  c34   , ud  a17   ;     end; comment a16 is changed and reset by c45.
     hr  s1          NRB   ;   reset input;
     arn d9    D     NTA   ;   if LRB ∧ inproc then stack (search);
     hv  c7          NTA   ;     comment array and switch has LRB;
     hr  s1                ;   return;


c35: arn(b1)   D           ; test decl: Raddr:= par 3;
b14: ca [last decl], hr s1 ;   if Raddr = last decl then return;
     ga  b14   , hv  c7    ;   last decl:= Raddr; goto stack;
a15: qq c33.19+8.29+2b.39, ;   action word for undeclared array
```

```
c36: hs  c13                ; spec1: comment array, switch, or unspec;
     arn(e1)   , hs  c10    ;   copy 1; Raddr:= last byte;
     grn(b7)          LZ    ;   search usage;
     bs  s511         LRB   ;   if found then cancel (entry);
     hv  c1                 ;   if number of indices = 0 ∨ -, array spec then
     hs  c4           NZA   ;     goto stack outnext;
     arn s     D            ;   if -, active then set active;
     ac (2b)   D   2        ;   Raddr:= number of indices;
     qq (b2)   t d11  -d12  ;   variables:= Raddr + 2 + variables;
     ps  c1-1  , hv  e3     ;   par 1:= par 1 + take array - formal array
                            ;   set return (stack out next 1); goto output;

c37: hs  c4             NZA ; spec 2: comment value;
                            ;   if -, active then set active;
c38: ps  c1-1  , hv  c13    ; spec 3: comment therest;
                            ;     set return (stack out next); goto copy 1



c39: hs  c13                ; copy code: copy 1:
     arn(e1)   , tk  -30    ;   CRcount:= CRcount + last byte x 2↑(-30);
     ac  e4                 ;
                            ;
a18: arn(e1)   t   1        ; loop code:
     hs  e2           LA    ;   Raddr:= next byte;
     hv  c2           LT    ;   if Raddr > 512 then goto outnext;
     ps  a18-1 , hv  e3     ;   set return (loop code); goto output;



                            ; forelem:
c40: hs  c6             LPB ; assign: if warning then output(par 3);

c41: pm  r                  ; simple for: warning:= f; goto outnext;

c42: hv  c2             IPB ; set warning: warning:= t; goto outnext;

s                           ;
```

[30.6.66]

[GIER Algol 4, pass 4, page 10]

[CONTROL TABLE: Each word holds up to four byte. The central input (c3:)
unpacks all four if the word is ,marked otherwise only the two first, it
leaves always the marks in RC where they are used to distinguish variants
of actions common to more input bytes.
The 4 bytes are usually used as:
output, action, stack value or extra output, where to count identifiers.
In the comments ++ indicates same meaning as input and ¨ indicates
extra output. Marks are given as A (,), B (f), C(f,), or N (none).]

```
d d2= i-1 [base table]              ; Input:        out,act,stack,count,marks

qq  84.9+c11.19                     ;  1 literal in: ++, literal, -, -, N
qq  85.9+c11.19                     ;  2    -    re: ++, literal, -, -, N
qq  86.9+c11.19                     ;  3    -    bo: ++, literal, -, -, N
qq  87.9+c11.19                     ;  4    -   str: ++, literal, -, -, N

d d7= i-d2 [input byte from here to d6 terminates trouble]

qq       c33.19+ 52.29+ 2b.39  , ;  5 decl simpin: -, declare  , ++, var, A
qq       c33.19+ 53.29+ 2b.39  , ;  6     -    re: -, declare  , ++, var, A
qq       c33.19+ 54.29+ 2b.39  , ;  7     -    bo: -, declare  , ++, var, A
qq  21.9+c45.19+  7.29+ 3b.39  , ;  8 decl label : tab:, decl lab, ++, loc, A
qq       c33.19+ 56.29+ 1b.39  , ;  9 decl own in: -, declare    , ++, own, A
qq       c33.19+ 57.29+ 1b.39  , ; 10     -    re: -, declare    , ++, own, A
qq       c33.19+ 58.29+ 1b.39  , ; 11     -    bo: -, declare    , ++, own, A
qq  28.9+c31.19+ 60.29+ 2b.39f , ; 12 decl arr in: begbnds in,decl arr, ++, par 3, C
qq  29.9+c31.19+ 61.29+ 2b.39f , ; 13     -    re: begbnds re,decl arr, ++, par 3, C
qq  30.9+c31.19+ 62.29+ 2b.39f , ; 14     -    bo: begbnds bo,decl arr, ++, par 3, C

qq       c26.19                     ; 15 end clean  : -, end clean, -, -, N
qq  17.9+c25.19                     ; 16 end block  : ++,end block, -, -, N
qq  14.9+c28.19                     ; 17 end bounds : ++,bounds   , -, -, N
qq  18.9+c24.19                     ; 18 end proc   : ++,end proc , -, -, N
qq  19.9+c24.19                     ; 19 endtypeproc: ++,end proc , -, -, N

qq   1.9+c17.19                     ; 20 begin      : beg block, begin, -, -, N
qq        c3.19                     ; 21 ;          : -, next, -, -, N
qq 172.9+ c2.19                     ; 22 do         : ++,outnext, -, -, N
qq 173.9+ c2.19                     ; 23 then st    : ++,outnext, -, -, N
qq 174.9+ c2.19                     ; 24 else st    : ++,outnext, -, -, N
qq 175.9+ c2.19                     ; 25 of st      : ++,outnext, -, -, N
qq 176.9+ c2.19                     ; 26 end case st: ++,outnext, -, -, N
qq   4.9+c39.19                     ; 27 code end   : beg code, copy code, -, -, N
qq  12.9+ c2.19                     ; 28 core code  : ++,outnext, -, -, N
qq   5.9+c28.19                     ; 29 corecodeend: ++,bounds , -, -, N
qq       c17.19+ 20.29       f , ; 30 endspec    : -, begin, specs, -, C

d d6= i-d2-1 [input bytes from d7 to here terminates trouble]

qq 177.9+c27.19                     ; 31 end call   : ++, start count, -, -, N
qq 178.9+c27.19                     ; 32 ] one      : ++, start count, -, -, N
qq 179.9+c27.19                     ; 33 ] more     : ++, start count, -, -, N
```

[30.6.66]

[GIER Algol 4, pass 4, page 11]

```
                                        ; input          : out,act,stack,count,marks

qq 180.9+c30.19                         ; 34 call param  : ++, count param, -, -, N
qq 181.9+c29.19                         ; 35 comma 1     : ++, count index, -, -, N
qq 182.9+c29.19                         ; 36 comma 2     : ++, count index, -, -, N
qq 183.9+c29.19                         ; 37 boundcolon  : ++, count index, -, -, N
qq  22.9+c23.19                         ; 38 beg call    : ++, counter out, -, -, N
qq   3.9+c23.19                         ; 39 beg func    : ++, counter out, -, -, N
qq  23.9+c22.19                         ; 40 [           : ++, beg subscr , -, -, N
qq       c14.19                       f ; 41 trouble     : - , trouble    , -, -, B

qq  36.9+c48.19+ 51.29+ 3b.39   ,       ; 42 dclparprno  : begpr no, par pr,++ , loc, A
qq  37.9+c18.19+ 48.29+ 3b.39   ,       ; 43      -  in  : begpr in, par tppr, ++, loc, A
qq  38.9+c18.19+ 49.29+ 3b.39   ,       ; 44      -  re  : begpr re, par tppr, ++, loc, A
qq  39.9+c18.19+ 50.29+ 3b.39   ,       ; 45      -  bo  : begpr bo, par tppr, ++, loc, A
qq  35.9+c48.19+  6.29+ 3b.39 f,        ; 46 decl switch : beg swch, par pr  , ++, loc, C
qq  40.9+c19.19+ 47.29+ 3b.39   ,       ; 47 decl procno : begpr no, no parpr, ++, loc, A
qq  41.9+c44.19+ 44.29+ 3b.39   ,       ; 48      -  in  : begpr in, nopartppr, ++, loc, A
qq  42.9+c44.19+ 45.29+ 3b.39   ,       ; 49      -  re  : begpr re, nopartppr, ++, loc, A
qq  43.9+c44.19+ 46.29+ 3b.39   ,       ; 50         bo  : begpr bo, nopartppr, ++, loc, A
qq  40.9+c19.19+  8.29+ 3b.39   ,       ; 51      -  un  : begpr no, no par pr, ++, loc, A

qq  76.9+c38.19+1023.29         ,       ; 52 spcsimp in  : form in   , spec 3, ++, -, A
qq  77.9+c38.19+1022.29         ,       ; 53      -  re  : form re   , spec 3, ++, -, A
qq  78.9+c38.19+1021.29         ,       ; 54      -  bo  : form bo   , spec 3, ++, -, A
qq  79.9+c38.19+1020.29         ,       ; 55      - str  : form st   , spec 3, ++, -, A
qq   9.9+c38.19+1019.29         ,       ; 56      -  la  : form la   , spec 3, ++, -, A
qq  68.9+c37.19+1018.29         ,       ; 57 specval in  : val  in   , spec 2, ++, -, A
qq  69.9+c37.19+1017.29         ,       ; 58      -  re  : val  re   , spec 2, ++, -, A
qq  70.9+c37.19+1016.29         ,       ; 59      -  bo  : val  bo   , spec 2, ++, -, A
qq  80.9+c36.19+1015.29       f,        ; 60 specarr in  : formarr in, spec 1, ++, -, C
qq  81.9+c36.19+1014.29       f,        ; 61      -  re  : formarr re, spec 1, ++, -, C
qq  82.9+c36.19+1013.29       f,        ; 62      -  bo  : formarr bo, spec 1, ++, -, C
qq  75.9+c38.19+1012.29         ,       ; 63 spcproc no  : form pr no, spec 3, ++, -, A
qq  72.9+c38.19+1011.29         ,       ; 64      -  in  : form pr in, spec 3, ++, -, A
qq  73.9+c38.19+1010.29         ,       ; 65      -  re  : form pr re, spec 3, ++, -, A
qq  74.9+c38.19+1009.29         ,       ; 66      -  bo  : form pr bo, spec 3, ++, -, A
qq  13.9+c36.19+1008.29         ,       ; 67      -  la  : form pr la, spec 1, ++, -, A
qq  11.9+c36.19+1007.29         ,       ; 68 un spec     : form un   , spec 1, ++, -, A
qq  10.9+c36.19+1006.29         ,       ; 69 spec gm     : form gm   , spec 1, ++, -, A

qq 184.9+c41.19                         ; 70 simple for  : ++, simple for, -, -, N
qq 185.9+c40.19+192.29          ,       ; 71 := for      : ++, forelem, whilelabel, -, A
qq 186.9+c40.19+192.29          ,       ; 72 step elem   : ++, forelem, whilelabel, -, A
qq 187.9+c40.19+192.29          ,       ; 73 while elem  : ++, forelem, whilelabel, -, A
qq 188.9+c42.19                 f       ; 74 while       : ++, set warning, -, -, B
qq 189.9+c42.19                 f       ; 75 end ass     : ++, set warning, -, -, B
qq 190.9+c40.19+193.29          ,       ; 76 :=          : ++, assign, prepass, -, -, A
qq 191.9+c40.19+193.29          ,       ; 77 first:=     : ++, assign, prepass, -, -, A
d d1= i-d2-1 [max interest]             ;
```

[30.6.66]

[GIER Algol 4, pass 4, page 12]

```
d15= i-1, d18= -d15          ; define stack base addresses
                             ; init stack:
a20: grn sc46          MC    ;   for i:= top core step -1 until stack base + 4 do
     bs  s511  , hv  a21     ;     store[i] MC:= 0; goto first byte is read;
     ps  s-1   , hv  a20     ;   comment end block;
c46: pa  3e4   t   2         ; start pass 4: max block level:= 2;
     qq (e1)   t   1         ;   stack top:= stack base;
     pi  128   , pp  d15     ;   inproc:= inhead:= active:= warning:=
     arn d19   D             ;     in trouble:= in block:= f;
     hs  e3                  ;   output (end pass);
     ps e20-c46,  hvn a20    ;   goto init stack;

b k=e31, i=0     ; load texts
i=e32            ;

d10: tbegin ends;
d16: tindices;   ;

e32=i            ;
e                ;

d17: qq [pass sum]           ;
d e22=k-e14, e47=j           ; set load parameters
b k=e23,i=0                  ; load segment word 6
i=6e21                       ;
     qq e16.9+1d17.19-e16.19+4.24+3.29+c46.39 ;
e                                             ;

e                            ; final end pass 4
s
```

```
b k=e22+e14, i=e16-e47, a27, b27, c34, d28 ; drum block head;
d i=e16                                    ;

[Specification of address variables. b1 - b7, see page 4
                        referred by:
b8h:  outrel            1c10, exc. from a13h
b9:   n                 1a5
b10:  ident act         c8, c9, c23, c26
b11:  byte              c10, 2c12, 7c12,1a8,3a14,1c20,c21,6c21,2c22,5c22,6c22,11c22
b12   entry             9c12, 2a6
b13   decl act          13c12
b14   stack ref         1a10, a13,
b15   L                 b15, c33
b16         unused
b17         unused
b18         unused
b19         unused
b20h  rel ref           12c12
b21   min ident         16c24, 6c34
b22   i                 c24, 1c24, 5c24, 10c24
b23   decl top          7c22, 14c24, 2c25, 2c26, 3c26, c30
b24   anon              7c24, 8c24, 11c24
b25   specrel           6c28, c32
b26   doperel           3c28
b27   n subscr         1c28 ]


[Predefinitions:]

d1=196    ; CR outputvalue
d2=404    ; Base for kind type. Added at output by outputdescr.
d3=87     ; Maximum value of input bytes which are treated by pass 5.
d6=190    ; Input value for := Used by output descr to test
d7=191    ;    -    -    - first:= for assign to procedure value
d9=471    ; Base for specifications in output
d11=12    ; Kindtype = undeclared
d15=445   ; Simple integr. Outputvalue used by take value and
          ;   take array.
d18=199   ; Take array. Output value
d19=48    ; Kind type = dope descr.
d20=27    ; Miminum input byte where two last bits gives type - 1.
d21=198   ; begin proc.
d22=132   ; output (take value int) - input (take value int)
d25=40    ; input value begproc no
d27=197   ; beg block
d28=96    ; undeclared.4
```

[The input table and the central logic of pass 5.

When entered at next the central logic inputs a byte and treats it in one of 4 ways depending on size:

1. byte = 0: CARRET action, return to next.

2. byte > 511 (i.e. negative): Jumps to the current identifier action. There are 3 possible actions on an identifier:
   1. It is declared, i.e. entered in the table with the current description given in the variable decl.
      This identifier action is set by begin block or begin proc.
   2. The table entry is checked for double declaration.
      This action is set by enddecl, endbounds, end proc, and end core code, and is explicitly performed by label colon
   3. The corresponding description is output from the table.
      This action is set by end head and core code.

3. byte > max interest. The byte is output, return to next.

4. byte ≤ max interest. The byte byte refers to the input table and is used as follows:
   If byte > min with type then byte : 4 is used as index in the table otherwise byte itself is used.
   The input table word is added to the byte and gives hereby a word in R in one of two formats:
   1. Declaration word (R positive):
      <kind type>.9 + <decl act>.19 + <how to get relative address>.29 + <marks to store in table>. marks.
      The variable decl is set to kind type.39 and curr block.33 is added to decl.
      The marks are saved in PC, relref is set to part 3, M is cleared and a jump to decl act is performed.
   2. Output and action word (R negative):
      qq <output value> + 512, hv <action>
      output value is output and a jump to <action> (via the table) is performed.]

[14.7.66]
[GIER Algol 4, pass 5, page 3]

[THE DECLARATION TABLE. Each identifier is represented by one word which is referenced by the pass 2 representation of the identifier and which holds the current valid declaration of that identifier.
Three main formats are distinguished:

Standard identifiers:
qq 0.0 + identpart.9 + Lpart.19 + Tpart.29 + chainpart.39, f;
identpart: pass 2 representation - 512. Only used during unstacking of a redeclared identifier.
Lpart: Representation of the std proc in pass 5 output. Is initialized to 0 and set to L before it is output first time.
L is initialized to e20 and decreased by 2 before each time it is used to set a Lpart.
Tpart: Number in identifier table. Used by finis pass 5
chainpart: Reference to identifier with next higher Tpart.
Used by finis pass 5.
These words are initialized by finis pass 4 after it has initialized:

Not used:
qq 0 f;
At each endblock or end proc all identifiers declared in that block are reset to this value.

All declared or used identifiers:
qq 1.0 + identpart.9 + relpart.19 + refpart.28 + blockpart.33 + declpart.39 (marks: see below)
identpart: As above.
relpart: Block relative address
refpart: 0 or reference to a word in the stack which holds
For arrays with knownno of subscr: Description of the dope vector
For procedures with parameters : The specificationlist.
For procedure values : Description of the procedure
block part: Block number
decl part: Kind and type
These words are set whenever an identifier is declared after d possible stacking of a declaration in an outer block.

DISTRIBUTION OF IDENTIFIERS: six primary formals are recognized by the marks and bit 0 as follows:

	Output:	bit0	marks	
(1)	Normal identifier: <base+decl part><relpart><blockpart>	1	0	1
(2)	Own or			
(2a)	Undeclared: <base+declpart><relpart>0	1	1	0
(3)	Not used: Is replaced by 1a and then output after alarm	0	0	1
(4)	Array with subscr: As 1 followed by dope descr (also as 1)or			
(4a)	Procedure with param: As 1 followed by specification list	1	0	0
(5)	Proc value: If following delimiter is:=or first:= then as (1)			
(5a)	else the word referenced by ref. part (as 1 or 4a)	1	1	1
(6)	Standard procedure: Lpart	0	1	1]

```
e16: qq [stackref block0.9+ min L .19+ std proc chain.39]   ; set by start and
     qq [silly, must be 0, see 1c33]                        ; fin pass 5

c1:  hs  c2                 ; copy 4: copy 1;
     hs  c2                 ;   copy 1;
     hs  c2                 ;   copy 1;

c2:  hs  c3                 ; copy 1:
     hv  e3                 ;   output (take byte); return


c3:  arn(e1)   t   1        ; take byte:
     hr  s1            NA   ;   Raddr:= next byte; return;

a1h: hv  e2    , ac  e4     ; treat CR: CR count:= CR count + 1;
     arn d1    DV           ;   output (out CR); return;

c5:  arn(s)    D            ; byte out: Raddr:= part 1 (store[s])
     hv  e3                 ;   goto output;

c4:                         ; end bound head:
bh:  pm  b6    , is[arr rel];   output (arr rel + 2);
     arn s2    D            ;     comment addr of coeff;
     hs  e3                 ;   output (array count)
     qq (b16)  , hs  c5     ;   output (kind type pt);
     cln -6    , ud  c9     ;     comment array type
     ck  -4    , hs  e3     ;   set descr;

c6:  pa  b16   , hh  c11    ; beg bounds: array count:= 0; goto next 1;

c7:  hs  c3    , qq i-1     ; copy code:
     hv  e3           NT    ;   for Raddr:= take byte while Raddr < 512 do
     tk  -30   , sc  e4     ;     output (Raddr);
     tk  30    , hs  e3     ;   CRcount:= CRcount - Raddr×2↑(-30);
     hh  c11                ;

c8:  pa  b10   V   c21      ; set check: ident act:= check decl; goto next;
c9:  pa  b10   t   c10      ; set descr: identact:= output descr; goto next;
a19: qq  d23   , hr  s1     ;   d23 is used from 8c12

[b1 - b4 contains the current (i.e. last used) relative address for the 4
 kinds of addressing]

                            ; referred by:
b1:  ps [local addr] t -1   ; 1c6, 13c24, 1c26, 6c26, b8
b2:  ps [var addr]   t -1   ; 6c26, 1c28, 2c28, 3c28, 7c28, b8
b3:  ps [own addr]   t -1   ; b8, 1c34
b4:  ps [form addr]  t  1   ; 5c26, 1c27, 5c27, b8
b5:  qq [currblock]  -1.33  ; curr block 14c12, a8, 10c22, 15c24, 3c27, 7c34
b6:  qq [decl=ident pt.9+relpt.19+refpt.28+blockpt.33+kindtypept.39];
                            ; 14c12, 4c25, 1c29, 6c6
b7:  qq [dopedescr=relptofdope.19+dopedescr.39] d19.39 ; 2c28,4c28  ;
```

```
c10: arn(b11)          ITA ; output descr: R:= set TA (table[byte]);

a2:  gt  b8    X       IRC ; normal out: outrel:= set RC (relpart(R)); swap;
     hv  a9            LRC ;   if LRC then goto proc val or std;

a3:  cln -6    , ck  -4    ; cont out: Raddr:= decl part of (M);
     ar  d2    DV      LTA ;   if NTA then
     arn(e1)   , hv  a8    ;     begin R:= same byte; goto undeclared end;
a4:                        ;   Raddr:= Raddr + base kind type;
b8:  hs  e3    , nc[outrel]; cont out 1: output (Raddr);
     arn s     D           ;   comment a4h executed from a13h;
     hs  e3                ;   output (outrel);
     cln -5    V       NRA ;   Raddr:=
     qqn       V           ;     if NRA then (- block part of (M)) else 0;
     ck  -5    , mt  r     ;
     hs  e3            LTA ;   if LTA then output (Raddr);
     hh  a10           NRC ;   if NRC then goto cont from stack;
a5:
b9:  bt [n]    t   -55     ; count output: n:= n + 1; if n = 10 then
     pa  b9    , ud  a7    ;     begin n:= 0; inf 1:= inf 1 + 1 end;
c11h:qq        , ps  i     ; next 1: set return (next);
                           ;   comment a return to next will do as next 1;
c12: pmn(e1)   X   1       ; next:
     hs  e2            LA  ;   Raddr:= next byte;
     ga  b11   V       NZ  ;   if R = 0 then goto treat CR;
     arn a17   , hh  a1    ;   byte:= Raddr;
b10: hv [ident act]    LT  ;   if Raddr > 511 then goto ident act;
b11: bs [byte] t d3        ;   if byte > of interest then goto output;
     hv  e3                ; unpack from intab: entry:= base in tab +
     bs (b11)  t d20       ;     (if byte > min with type then byte:4+d23 else byte);
     ck  -2    , ar  a19   ;   R:= byte + set QC (intab [entry]);
     ga  b12   , arn(e1)   ;   if NT then set declaration:
b12: ar [entry]t  d4   IQC ;     begin
     hv  a6            LT  ;     set PC; comment marks from in tab [entry];
     cl  10    , gt  b20   ;     decl act:= part 2 (R);
     ga  b13   X       IPC ;     relcount:= part 3 (R);
     ar  b5    , gr  b6    ;     decl:= part 1 (R) shift (-30) + curr block;
b13: hvn[decl act] X       ;     R:= R × 2↑10; M:= 0;
                           ;     goto decl act
                           ;     end;
a6:  mb  511   D           ;   Raddr:= bits (1, 9, R);
     hs  e3            NZ  ;   if Raddr ≠ 0 then output (Raddr);
     hh (b12)              ;   goto right half (in tab [entry]);


a7:  qq (2e4)  t   1       ;   comment executed from 1a5
```

```
a8:  ar  a18    , ar  b5     ; undeclared:
     gr (b11)   , hs  e5     ;   table [byte]:= R + curr block + undecl;
     hv  c10    , qq  sd5    ;   mess(‡<<undeclared‡>, 0, 1);
                             ;   goto output descr;

a9:  hh  a14           NTA   ; proc val or std: if NTA then goto std proc;
     hs  c3     , qq  i-1    ;   for Raddr:= take byte while R = 0 do
     qq (e1)    V   -1  NZ   ;     treat CR
     arn a17    , hh  a1     ;   reset input;
     nc  d6     , ca  d7     ;   if Raddr = v:= v Raddr = vfirst := then
     hv  a3           IRA    ;     begin RA:= f; swap; goto cont out end;
                             ;
a10h:cln -11    , cln -9     ; cont from stack:
     ck  -1     , ga  b14    ;   stack ref:= ref part of (M);
a11:                         ;
b14: pmm[stack ref]t d8 IRC  ; next word: M:= set RC (stack [stack ref]);
     hv  a2     X     NRA    ;   if -, RA then begin swap; goto normal out end;
a12: cln -5     , ck  -5     ; next spec: Raddr:= bits (35, 39, M);
     hv  a13          LZ     ;   M:= M : 2↑5;
     ar  d9     D            ;   if R ≠ 0 then
     ps  a12-1  , hv  e3     ;     begin Raddr:= Raddr + basespec;
                             ;       set return (next spec); goto output end;
a13: hv (b14)   D d17   NRB  ;   if NRB then more words:
                             ;     begin stackref:= stack ref+1; goto next word end
                             ;   goto count output;
a14h:hv  a5     , udn a4     ; std proc:
     ps  a4     , hr  s1     ;   if outrel ≠ 0 then
                             ;     begin set return(contout 1+1); return end;
b15: it e20[L] t   -2        ;   relpart of table [byte]:= L:= L - 2;
     pt (b11)   , hv  c10    ;   goto output descr;


                             ; Constants:
a15: qq 1.0 + 31.33          ;   block part mask
a16: qq       1.33           ;   unit for block count
a17: qq             1.39     ;
a18: qq           d11.39     ;   undeclared
```

```
c20: hs  c3                  ; label colon:
     ga  b11                 ;   byte:= take byte;

c21: arn(b11)  , gt  b       ; check decl: R:= table[byte]; arr rel:= part 2(R)
b16: qq[array count] Vt1NC   ;   comment for use by end bound head;
     cl  -6    X       NB    ;   if array or proc then array count:= array count + 1;
     ca  d28   , hs  e5      ;   if decl part (R) = undecl then mess(‡<+decl›,0,1);
     hh  c11   , qq  sd12    ;   goto next 1;


c22:                         ; declare:
b20h:ga  b6  , ud [rel ref];   ident part := Raddr; rel part:=
     it  s     , pt  b6      ;     rel addr[rel ref] + rel incr[rel ref];
     arn(b11)  V       IRC   ;   R:= set RC (table [byte]);
a20: qq (3e4)  t   1         ;   if R = 0 then
     pm  b6    V       LZ    ; new declaration: table [byte] MPC:= decl;
     pm  a15   , hh  a21     ;   else
     gm (b11)          MPC   ;     begin
a21: hh  c11   , cm  b6      ;     if block part and bit 0 (table[byte]) ≠
                             ;       block part and bit 0 (decl) then
     gr (b23)  V   1   MRC   ;     redeclaration:
     arn a18   V             ;       begin decl top:= decl top + 1; inf2:= inf 2 + 1;
     ps  a20-1 , hvn c30     ;       stack[decl top] MRC:= R; R:= 0;
                             ;       set return (new declaration); goto check stack
     ar  b5    , ar (e1)     ;       end;
     gr (b11)          MA    ;     double declared:
     ps  c12-1 , hv  c12     ;       table[byte] MA:= byte + curr block + undecl;
                             ;     end
                             ;   set return (next); goto next;
```

```
c23: pa  b10   t   c21     ; end proc: ident act:= check decl;
     hs  c2                ;   copy 1;

c24: pa  b22   V   e20+1   ; end block:
a22: grm(b22)         MB   ;
b21: arn[min ident], pm a15;   for i:= max ident step -1 until min ident+1 do
b22: cm [i]    t   -1      ;     if block part (table[i]) = curr block ∧
     hv  r-1               ;       bit 0(table[i]) = 1 then table[i] MB:= 0;
     nc (b22)  , hv  a22   ;
b23: arn[decl top] d8  IRC ; un stack declaration:
     ga  b24   V       LT  ;   for R:= set RC [stack[decl top]) while LTVLRC do
     ga  b24   , hv  a23   ;     begin
b24: gr [anonymous]    MRC ;     table [identpart (R) + (if LT then 0 else 512]
     hv (b23)  D   -1      ;       MRC:= R;
a23: hv (b24)  D  512  LRC ;     decl top:= decl top - 1
                           ;     end;
     gt  r     , pp  -1    ; block stop now in R: spectop:= part 2(R)
     qq (b23)  t   -1      ;   decl top:= decl top - 1; R:= in pos block (-1);
a24h:srn a16   , ac  b5    ; set block no: curr block:= curr block + R;
     ac (b21)  , hv  c12   ;   table [min ident]:= table [min ident] + R;
                           ;   goto next;

c25: qq  d21   , hs  c5    ; beg proc: byte out (beg proc);
     srn d25   D           ;   out (byte - beg proc no)
     ar (e1)   , hs  e3    ;   ref part:=
     is (b23)              ;     decl top - decl base + 2;
     arn sd14  D           ;
     ck  -19   , ac  b6    ;
c26: pan b10   X   c22     ; beg block: ident act:= declare; decl top:= decl top+1
     gm (b23)  t   1   MA  ;   stack [decl top] MA:= 0
     it  p     , pt (b23)  ;     + in part 2 (spec top);
     hs  c30   , qq  1     ;   check stack; form addr:= 1;
     gs  b4    , hs  c3    ;   local addr:= 0; var addr:= take byte;
     ga  b2    , pa  b1    ;   copy 1; comment base work;
     hs  c2    , qq  c12-1 ;   set return (next);
     arn a16   , hh  a24   ;   R:= inpos block (1); goto set block no;


                           ; take formal:
c27: qq  d15   , hs  c5    ;   byte out (simple formal);
     arn(b4)   D   1       ;   form addr:= form addr + 1;
     hs  e3                ;   output(form addr);
     srn b5    , tk  24    ;   output (-curr block);
     hs  e3    , qq  c12-1 ;   set return (next)
     qq (b4)   V   -1  LPB ;   if NPB then array specification:
     ps  d16   , hv  c28   ;     begin set return (take array); goto array decl end;
     arn d22   D           ;   form addr:= form addr - 1; Raddr:= last byte +
     ar (e1)   , hv  e3    ;     take value; goto output;
```

```
c28: hs  c3                  ; array decl: n subscr:= take byte;
     ga  b27   , sc  b2      ;   var addr:=
     it (b2)   , pt  b7      ;     rel part of dope:= doperel:=
     it (b2)   , pa  b26     ;     var addr - n subscr;
     ck  16    , ar  b7      ;   stack [spec top - 1] MB:=
     gr  p-1           MB    ;     dope descr + inpos block no (n subscr);
     gp  b25   , pp  p-1     ;   spec ref:= spec top; spec top:= spec top - 1;
     qq (b2)   t   -1        ;   var addr:= var addr - 1;
c29:                         ;
b25: arn[spec ref] Dt d17    ; par pr decl:
     ck  -19   , ac  b6      ;   ref part := spec ref - stack base - 1;

c30: is (b23)  , it  s-512   ; check stack:
     bs  p-512 , hr  s1      ;   if decl top < spec top then return;
     ps  a25   , hv  e5      ;   mess (<<stack>, 2, 0);


c31: qq  d18   , hs  c5      ; take array: byte out (take array);
b27: qq [n subscr], hs c5    ;   byte out (n subscr);
b26: qq [doperel], hs c5     ;   byte out (doperel);
a25=i-1 [mess descr]         ;   goto next 1;
     hh  c11   , qqn e34     ;
d16=c31-1[ret. to take arr];


a26: bs  s511  , hv  a27     ; pack spec: if spec pos > 0 then
                             ;     begin spec top:= spec top - 1;
     pp  p-1   , gm  p       ;     store[spec top]:= 0;
     hs  c30   , qq  -30     ;     check stack; spec pos:= -30
                             ;     end;
a27: tk  s     , ps  s5      ;   stack [spec top] MC:= stack [spec top] -
     sc  p     V       MA    ;     R × 2↑spec pos; spec pos:= spec pos + 5;
                             ;   goto next spec;
c32: gp  b25   , ps  5       ; specs: spec ref:= spec top; spec pos:= 5
     hsn c3    X             ;   R:= take byte;
     hv  a26           LT    ;   if R < 0 then goto pack spec;
     acn p             MC    ;   stack [spec top] MC:= stack [spec top];
     qq (e1)   t   -1        ;   reset input;
     ps  c11-1 , hv  c11     ;   set return (next); goto next;

c33: it (b15)  , pt  e16     ; fin pass 5: min L:= L;
     arn 1e16  , hh  e29     ;   R:= descrp pass 5 segm 2; goto new segm
```

[input table: Entries starting qq - are declaration words (see page 2),
the others are output action words]

```
d4 = i-1 [table base]                ; Input value        comment

qq 511.9+197.9, hv  c26              ;  1 begblock
qq 510.9+213.9, hv  c33              ;  2 end pass
qq 509.9+214.9, hv  c2               ;  3 beg func
qq 508.9+210.9, hv  c7               ;  4 begcode
qq 507.9+211.9, hv  c8               ;  5 end core code
qq  -6.9+ 20.9+ c12.19+ b1.29 f      ;  6 decl switch
qq  -7.9+ 16.9+ c12.19+ b1.29 f      ;  7 decl label
qq  -8.9+d11.9+ c12.19+ b1.29 f      ;  8 decl pr undef

qq  -9.9+ 24.9+ c12.19+ b4.29 f      ;  9 form label
qq -10.9+  8.9+ c12.19+ b4.29 f      ; 10 form gener
qq -11.9+d11.9+ c12.19+ b4.29 f      ; 11 form unsp
qq 500.9+212.9, hv  c9               ; 12 core code
qq -13.9+ 28.9+ c12.19+ b4.29 f      ; 13 form switch

qq 498.9+203.9 , hv   c8             ; 14 endbounds
qq -15.9+  0.9+  c9.19               ; 15 end head    15, 16 and 20 are decla-
qq -16.9+  0.9+  c8.19               ; 16 enddecl     ration words to prevent
qq 495.9+204.9 , hv  c24             ; 17 end block   output action and decl
qq 494.9+205.9 , hv  c23             ; 18 end pr      is free at these inputs.
qq 493.9+206.9 , hv  c23             ; 19 end tppr
qq -20.9+  0.9+ c32.19               ; 20 specs
qq 491.9+207.9 , hv  c20             ; 21 labcolon
qq 490.9+208.9 , hv   c2             ; 22 beg call
qq 489.9+209.9 , hv   c2             ; 23 [
qq 488.9+215.9 , hv   c4             ; 24 end bound head
d23=i-d4-7                           ;
[table entry for the following input bytes is computed as
 d4+d23+byte:4]
qq -28.9+  1.9+  c6.19               ; 28 beg bounds  kind tp pt is array type
qq -32.9+103.9+ c25.19               ; 32 beg switch  32, 36, 40 must be together
qq -36.9+103.9+ c25.19+ b1.29 f,     ; 36 beg par proc  proc type is output as
qq -40.9+103.9+ c25.19+ b1.29 f,     ; 40 beg proc      input - beg proc no
qq -44.9+ 32.9+ c12.19+ b1.29 f      ; 44 decl tppr
qq -48.9+ 36.9+ c29.19+ b1.29        ; 48 declpartppr
qq -52.9+ 40.9+ c12.19+ b2.29 f      ; 52 decl simpel
qq -56.9+ 40.9+ c12.19+ b3.29  ,     ; 56 decl own
qq -60.9+ 44.9+ c28.19+ b2.29        ; 60 decl array
qq -64.9+ 44.9+ c27.19+ b2.29        ; 64 take array
qq -68.9+ 40.9+ c27.19+ b4.29 f      ; 68 take value
qq -72.9+ 52.9+ c12.19+ b4.29 f      ; 72 form tp pr
qq -76.9+ 56.9+ c12.19+ b4.29 f      ; 76 form simpel
qq -80.9+ 60.9+ c12.19+ b4.29 f      ; 80 anon array
qq 428.9+468.9 , hv   c1             ; 84 literal
```

```
d8=i-1 [Define stack base]          ;
d10=-d8+1, d14=-d8+2, d17=-d8-1     ; other stack addresses

[Start pass 5 is overwritten by the stack]

c34: hs  c3                  ; start pass 5:
     ga  b3    , hs  c3      ;   own address:= take byte
     gr  e16   , hs  e3      ;   stack ref bl 0:= R:= take byte; output (R);
     hs  c3                  ;   std proc chain:= take byte;
     ck  10    , ac  e16     ;   min ident:= Raddr:= take byte
     hs  c3                  ;
     ga  b21   , pp (b21)    ;   spectop:= p:= min ident;
     bs  p-d8-510, pp510d8   ;   if spec top - stack base > 510 then
     it  p     , bs d24      ;     spectop:= stack base + 510;
     ps  a25   , hv e5       ;   if spectop < last word pass 5 then
                             ;     mess(<<stack>, 2, 0);
     ar  b5    , gr (b21)    ;   table[min ident]:= R + curr block;
     qq  d27   , hs  c5      ;   byte out(beg block);
     hv  c26                 ;   goto beg block


b k=e31, i=0                 ; Load texts
i=e32                        ;
d5:  tundeclared;            ;
d12: t+ decl.;               ;
e32=i                        ;
e                            ;

d24:  qq [pass sum]          ;
d e22=k-e14, e47=j           ; set load parameters
b k=e23, i=0                 ; load segment word 7
i=7e21                       ;
qq e16.9+1d24.19-e16.19+5.24+1.28+c34.39;
e                                       ;

e [final end pass 5]                    ;
s                                       ;
```

[17.10.1967]
[Gier Algol 4, pass 5 std. proc descr., page 1]

[This code is taken in to 1e16 as a segment.
It sets up the standard identifier description table to pass 6:
The chain of standard identifiers set up by pass 4 is followed
starting with the address left in bits 30 - 39 of e16 by start pass 5
and ending when a chain 0 is encountered.
All words which have been used (Lpart ≠ 0) are stored in a list
of used std identifiers as qq <Lpart>.9 + <Tpart>.19.
The next segment is now taken to core; it contains section 2 of the
standard procedure library.
Each of the above words are then treated using Tpart as index in section 2
of the standard procedures and Lpart as address of where to store
the description in the pass 6 table.
Descriptions which refer to a proper standard procedure (NTR part ≠ 0)
are further used to build up a table of tracks to be encluded in
the generated code. The track no entered in these descriptions is
the position in the table of tracks of the entry track number for
the procedure.
Descriptions which refer to std variables (NTRpart = 0) will instead
of entry track get the relative address of the variable as if it
were in block 0.
Finally the track list is output.

Format of an entry in standard procedures section 2:
For a procedure:
qq <FTR>.9+<NTR.14+<ETR>.19+<code ref>.23+<rel>.29+<ref>.33+<kind type>.39
qq <spec n>.9+<spec n-1>.14+<spec n-2>.19 ...
For a variable:
qq <Address relative to displ block 0>.9+<code ref>.23+1.29+<ref>.33>+
<kind type>.39
qq <specn>.9 ...

FTR: first track of the procedure relative to first track of section 3
ETR: entry track relative to FTR.
NTR: number of tracks
rel: entry address on track ETR.
ref and kind type are references to pass 6 tables

Format of an entry in the pass 6 table:
For a procedure:
L: qq <ref>.9+<entry track>.19+<rel>.27+<code output>.33+<kind type>.39
L+1: qq <spec n>.9 ...
For a variable:
L: qq <ref>.9+<block 0 rel>.19+1.27+<kind type>.39
L+1 qq <specn>.9...]

```
b k=e22+e14, i=1e16-e47, a12, b10, d9 ; drumblock head
I=1e16                                ;

d4=200 ; max no of tracks allowed
d5=50  ; max no of std proc references alowed

a:   grn e20    , arn e16   ; start pass 5 std proc descr:
     ck  10     , ga  e20   ;   min L:= part 2 (e16);
     tk  20     , ga  b2    ;   chain:= part 4 (e16); i := 0;
     it  e20-d5-d5          ;   if min L < top L < 2×max allowed std procs then
     bs (e20)   , hv  a3    ; error: mess(≠<std.procs.≠, 2, 0);
a1:  ps  d7     , hv  e5    ;   else goto start chain;

a2:  arn(b2)    , ck  -10   ; loop chain: R:= store [chain];
     ga  b2     , tk  20    ;   chain := chainpart (R);
     nc  0                  ;   if L part (R) ≠ 0 then
b1:  gr  d1[i] t   1        ;     begin i:= i + 1; used list [i]:= R shift 10 end;
a3:                         ; start chain:
b2:  ncn[chain], hv  a2     ;   if chain ≠ 0 then goto loop chain;
     grn  d2    , hhn e29   ;   tracks[0]:= 0; goto next segment;
d3:  ppn 0      , ud  b1    ;     comment will come back here;
                            ;   i:=i+1; used list [i]:= 0; i:= 1; t:= 0;
                            ;   p:= number of std proc tracks:= 0; entry := -2;
a4:  pmn d1[i] X   1        ;   for R:= used list[i] while R ≠ 0 do
     hh  a8              LZ ;   begin i:= i + 1;
     ga  b5     , ck  10    ;   L:= part 1(R); T:= part 2(R);
     ga  b4                 ; loop t:
a5: b10: qqd8[entry] t 2    ;   entry:= entry + 2;
b3:  it [t]     t   1       ;   t:= t + 1 if t < T then
b4:  bs [T]     , hv  a5    ;     goto loop t;
     is (b10)   , arn s1    ;   pass 6 table[L+1] := std proc descr [entry + 1]
b5:  is [L]     , gr  s1    ;   R:= std proc descr [entry];
     arn(b10)   , ga  b6    ;   etr:= FTR:= part 1(R);
     ga  b8     , cl  34    ;   pass 6 table kind type [L]:= bits (34, 39, R);
     ck  16     , cl  -4    ;   pass 6 table ref [L]      := bits (30, 33, R);
     ck  -18    , cl  -6    ;   pass 6 table rel [L]      := bits (24, 29, R);
     ck  12     , cl  -4    ;   pass 6 table code ref [L] := bits (20, 23, R);
     ck  10                 ;   marks [L]:= 0;
     gr (b5)             M  ;   etr:= etr + bits (15, 19, R);
     cln -5     , ck  -5    ;   Rpos4:= bits (10, 14, R);
     ac  b8     , cln -5    ;   if R = 0 then
     hv  a6              NZ ;     begin
                            ;     not reference to std proc on tracks:
     ac (b5)             MA ;     marks [L]:= 2;
     nt (e16)   , it (b8)   ;     pass 6 table block 0 rel [L]:=
     pt (b5)    , hv  a4    ;       etr + rel stack ref 0;
                            ;     end
                            ;   else
a6:  ga  b7                 ;     begin
a7:                         ;     for Rpos4:= Rpos4 - 1 while Rpos4 ≥ 0 do
b6:  arn[FTR]   Dt  1       ;       begin
     it (pd2)   , bs (b6)   ;       FTR:= FTR + 1;
     pp  p1     , gr  pd2   ;       if FTR > tracks [p] then
b7:  ncn[Rpos4]t   -32      ;         begin p:= p + 1; tracks [p]:= FTR end
     hv  a7     , qqn d6    ;       end;
d7=i-2[address of message] ;
```

```
     it  p     , qq (b8)    ;     pass 6 table entry track [L]:=
     nt (pd2)  , it (b8)    ;       p + etr - tracks [p]
     pt (b5)   , hv  a4     ;     end
a8h:                        ;   end for R;
b8:  qq [etr]  , is (e20)   ;   marks[minL - 2]:= 1;
     grn s-2           MB   ;   output (p+1); run track:= run track - p + 1;
     arn p1    D          ;
     sc  13e4  , hs  e3     ;
     arn(e16)  D          ;   output (rel sr 0);
     hs  e3               ;
     arn(13e4) D   2      ; modify entry track so that first track will
     ck  -10              ;   will be run track:
a9:                         ;   j:= top L + 2;
b9:  pm  e20[j]t   -2       ;   for j:= j - 2 while marks [j] ≠ 1 do
     ac (b9)           NC   ;     pass 6 table entry track [j]:=
     hv  a9            NB   ;     pass 6 table entry track [j] + run track;

                            ; output track list:
a10: bs  p511  , hhn e29    ;   for p:= p step -1 until 0 do
     arn pd2   , hs  e3     ;     output (tracks [p]);
     pp  p-1   , hv  a10    ;   R:= 0;   goto next segm;

d1:  qq [used list [0];
d2=d1+d5 [tracks[0]   ;
d8=d2+d4 [stdproc descr[-2];

b k=e31, i=0        ; load text
I=e32               ;
d6: tstd.procs.;    ;
e32=I               ;
e                   ;

d:   qq [pass sum] ;

d e22=k-e14, e47=j ; Set load parameters;
b k= e23, i=0      ; Load segment word 8 and 9
I=8e21             ;
qq 1e16.9+1d.19-1e16.19+1.20+a.39f  ;
qq 2d8.9+63.19+     1.20+ d3.39   f,; length 63 is for debugging, can stay in;
e                  ;

e [final end]      ;
s
```

```
d e49=4      ; tape number := 4;

[After i follows STOPCODE, SUMCODE and a sum character]
ia T3
s
```